GENERATING BOMBER ROUTES FOR THE DELIVERY OF GRAVITY WEAPONS

THESIS

Gregory J. Ehlers
Captain, USAF

AFIT/GOR/ENS/98M-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/98M-11

GENERATING BOMBER ROUTES FOR THE DELIVERY OF GRAVITY WEAPONS

THESIS

Gregory J. Ehlers
Captain, USAF

AFIT/GOR/ENS/98M-11

The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the US Government

AFIT/GOR/ENS/98M-11

GENERATING BOMBER ROUTES FOR THE DELIVERY OF GRAVITY WEAPONS

THESIS

Presented to the faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Gregory J. Ehlers, B.S. Mathematics,

Captain, USAF

MARCH 1998

THESIS APPROVAL

STUDENT: Gregory J. Ehlers, Captain, USAF CLASS: GOR 98M

THESIS TITLE: GENERATING BOMBER ROUTES FOR THE DELIVERY OF GRAVITY WEAPONS

DEFENSE DATE: 6 March 1998

COMMITTEE:	NAME/DEPARTMENT	SIGNATURE
Advisor	Lt Col James T. Moore/ENS	James T. Moore
Reader	Lt Col J. O. Miller/ENS	J. O. Miller

ACKNOWLEDGEMENTS

I would like to thank the people who helped me successfully complete this thesis. First, thank you to Maj Mark Gallagher and Maj Elliot Fair at USSTRATCOM for their assistance, suggestions and provision of the information on the thesis process that guided me through. I would also like to thank Maj Tom Reid for the help with the math formulas needed to make the true backtracking generator possible. Of course, I can't give enough thanks to Lt Col James Moore and Lt Col J.O. Miller for their leadership, understanding and guidance. I have a great deal of appreciation for the assistance given to me with every aspect of this thesis by my roommate and classmate Lt Shawn Miller. Lastly, I would like to thank my family, Lisa, Zach and Briana for their support and the willingness to make the sacrifices that were required in order for me to complete this thesis.

Table of Contents

List of Figures

List of Tables

ABSTRACT

United States Strategic Command (USSTRATCOM) is updating the Arsenal Exchange Model (AEM) that it uses for allocating strategic weapons to targets. AEM does not model the range constraints on bomber routes. These routes are used to deliver gravity munitions. In order to include these constraints in the new method, the Weapons Allocation Model (WAM), a pool of routes that satisfy the range constraints needs to be created. The number of possible routes to consider is extremely large and it is not possible to enumerate them all. Therefore, the method to form the pool of routes needs to reduce the number of routes considered and selected to a manageable number. This research examines several methods for generating routes in an attempt to find the best method. All of these methods use implicit enumeration to create a pool of routes. The best method uses a combination of distance and flight direction to restrict the number of targets available to generate a route. However, the reduction provided by this method is not large enough and preprocessing the target database is also used. The method is tested against sample problems of several sizes to find the best way to use the method to generate routes in a problem of comparable size to the USSTRATCOM problem.

1. Introduction

Background

At the present time the United States Strategic Command (USSTRATCOM) is developing a new linear programming model to replace the Arsenal Exchange Model (AEM). AEM is used to make nuclear weapon to target assignments for both Blue-on-Red and Red-on-Blue analyses. The new model, called the Weapon Assignment Model (WAM), will use recent advancements in computer technology to improve the weapon assignment process. In light of these new improvements, USSTRATCOM is also improving other aspects of the weapon assignment process. One such aspect is improving the modeling of bomber routing--the focus of this research.

Currently in AEM, bomber range constraints are not accounted for in the optimization, but targets are swapped after the initial assignments to ensure each bomber load is within a very large rectangle. The rectangle is usually kept unrealistically large to avoid over constraining the problem and to minimize the swaps that move the solution away from the optimal solution. After AEM has completed the bomber to target assignments, depending upon the analysis task being accomplished, analysts may run OPUS (Operation Research Concepts Applied Planning Utility System) to build automated flight plans. The advantage of OPUS is that the individual routes for a selected series of targets can be generated in a totally automated fashion. The disadvantage is that targets for a selected bomb type are not sorted into logical sorties. For example, two Russian bombers may each be assigned targets on both the east and west coasts, rather than dividing the targets into a set on each coast. In addition to the

problem of selecting among the bomber targets for individual sorties, the bomber routes generated by OPUS are not realistically constrained by fuel or air defenses. While the routes generated by OPUS have limitations, as recent as two years ago no effort was made to find viable routes when using AEM for analysis.

For the actual United States nuclear war plans, military planners take target to bomber assignments and develop flight routes with the Automated Routing and Maintenance System (ARMS). ARMS is an event-oriented (discrete-event) simulation model. The planners use ARMS to select routes for each bomber to maximize the probability of arrival. Running ARMS and building bomber routes requires user interaction and is extremely time consuming. While the resulting ARMS routes are effective and very detailed, the detail provided is not required for the typical analysis task. In addition, the files necessary to use ARMS to support Red-on-Blue routing are not currently available.

The analysts at USSTRATCOM envision that WAM will select routes from a large database of previously built routes. In the optimization, WAM will only select targets if an available route exists. When the route is stored, the probability of arrival of the bomber at each target will also be stored. Therefore, WAM can optimize the damage expectancy (probability of arrival times probability of damage) in making weapon to target assignments. This presents two major advantages over AEM. First, the optimized solution will not have to be adjusted to obtain realistic routes, and second, a more realistic probability of arrival, rather than a simple probability of arrival planning factor, is used in the optimization.

The goal of this research is to adapt or develop an approach that will generate good bomber routes given a set of gravity-bomb targets. The pool of routes will very likely overlap and provide various routes to a single target. These routes should be worth evaluating with a detailed routing model, such as ARMS. One possible way to start building a pool of routes would be to run the optimization without bomber route constraints, and then run the approach from this research to identify potential routes through the targets assigned gravity weapons. Once some routes are developed, the optimization, WAM, could be run again and the reduced prices examined to identify targets for which a bomber route was available that would improve the objective function most. One objective of WAM is to maximize the damage expectancy. Therefore, routes with the largest reduced costs will be selected by WAM to be attacked with a gravity weapon. After adding those new potential targets to the bomber target pool, the method from this research could be applied again to find additional bomber routes to add to the pool of routes.

Range Constraints

Every bomber has a limited range. These bomber range constraints can be looked at several ways. One way would be to consider the amount of consumable fuel remaining. Another way would be to consider range as a function of time remaining given a fuel amount and a consumption rate. And lastly, range could be viewed as the flying distance available given the current fuel level and fuel consumption rate. All are essentially the same and with proper manipulation, the problem of generating feasible routes could be formulated using any of the above approaches. The flying distance is computationally easier to model and is the way this research models a bomber's range.

Like range constraints, there is more than one way to define the distance between two targets. Typically aircraft fly what is known as the great circle route because it represents the shortest route between two points on a sphere. The great circle between two points on the surface of the earth represents the arc of a plane intersecting the sphere (the earth) at its center and the two points. This research uses an equation from an Air Mobility Command (AMC) Excel spreadsheet for finding the great circle distance between all targets.

Range Constraints Versus Footprint Constraints

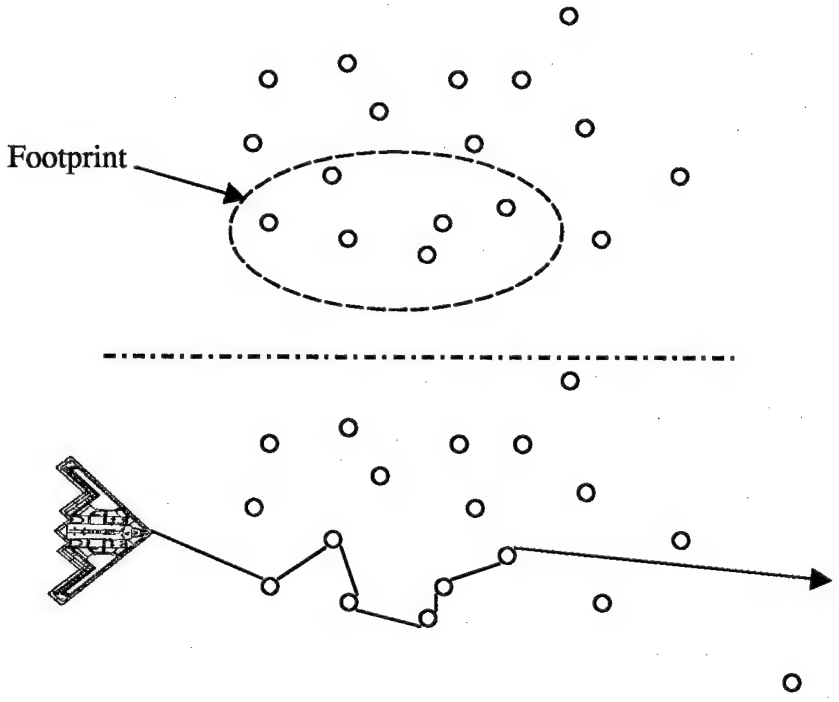


Figure 1-1. Comparison of Missile Footprint and Bomber Route

WAM will also assign other strategic weapons, like Intercontinental Ballistic Missiles (ICBM), to targets. One ICBM can carry several warheads. Each of these warheads can be assigned to individual targets, but there are limitations. Targets must be within a specific geographic distance of each other because the energy available to

disperse the warheads is limited. The area defined by this geographic distance is called a footprint. A bomber can also strike many targets, but which targets are attacked by a given bomber are also limited. In this sense, the bomber range constraint problem is similar to a missile footprinting problem that has already been researched [4]. Both problems involve determining which subset of targets from the larger set of targets is feasible for the associated weapons system. Unfortunately, the differences in how the subsets are constrained prevent the same model from working on both problems. Now, because of the large number of possible missile footprints, it is not computationally feasible to consider all of them in a linear program like WAM [2]. This will be true for the bomber routes. That is to say, it may not be feasible to consider all possible bomber routes.

Problem Definition

Bombers can attack many targets. In order to assign targets to a given bomber, bomber range constraints need to be considered. The order in which a bomber strikes targets will determine a route. Many routes between targets scattered over large distances will not be feasible. Simple enumeration of all possible routes based on the entire target set and comparison to a maximum range distance is not acceptable because it is too time-consuming and extremely inefficient. As will be seen in Chapter 3, the possible number of routes is greater than 10^{18}. For an approach to be considered efficient, the number and complexity of the calculations should be minimized. Not all routes that satisfy a given range constraint are desirable. There may be other criteria that can be used to further reduce the number of routes generated. The goal is to produce a good set of routes that can be used in WAM to optimize weapon to target assignment.

Scope

This thesis concentrates on generating bomber routes. However, since the number of possible routes is quite large, range constraints and other constraints are also applied. For example, there is a possibility that the method or heuristic selected may include the effect of enemy air defenses or some aspect of gravity weapon deployment. The reason for these type of inclusions would be to reduce the number of routes generated and decrease the time required to generate the routes.

Limitations

The designation of a particular facility, place, or building as a target for a nuclear weapon is extremely sensitive information and is not necessary for this research. A test set of targets will be needed in order to test the approach or approaches used to generate routes. The targets' location will be assigned randomly. The method and details used to generate these targets and the locations for each target will be described in more detail in Chapter 4.

The number of possible targets that can be assigned to a nuclear weapon is in the thousands, while the number of available nuclear weapons is small. It has already been stated that one way to reduce the number of targets that need to be placed on a bomber's route is to use WAM to assign gravity weapons to targets. This research will assume this has been done and the number of possible bomber targets has been reduced.

Format

Chapter 2 provides a more detailed explanation of the bomber routing problem and develops some of the concepts needed to understand the method used to solve the

problem. The next chapter compares the bomber routing problem to traditional Vehicle Routing Problems frequently discussed in Operations Research literature. Several different approaches to route generation are proposed and explored in Chapter 4. The method used to generate routes evolves from these approaches. This evolution process and the methods are also described in Chapter 4. Chapter 5 compares the results of the different approaches, develops an alternate approach and selects an approach based on the comparisons made earlier in the chapter. Finally, Chapter 6 provides recommendations for future research and suggests some potential improvements to the method for generating routes that performs the best.

2. Background

Introduction

Chapter 2 presents additional background information and assumptions for the bomber routing problem. This information is used in Chapter 4 where the methods for generating bomber routes are developed. The last section in this chapter explains the method used for finding the distance between the targets.

Current Bomber Routes

Before an approach is developed to generate bomber routes, there are a few aspects of bomber operations that are worth mentioning. The first is that because of the current world view on the use of nuclear weapons, the location of U.S. bomber bases, and the geometry associated with global travel, most bomber routes begin at their respective bases and fly north over the Artic Circle. This is related to the great circle distance mentioned in Chapter 1. Therefore, most of the refueling locations are to the north of the target country. Second, after dropping a nuclear gravity weapon, a bomber has to continue flying on basically the same heading as it was on when it dropped the weapon. In other words, a bomber does not turn around and fly directly over the area where a nuclear gravity weapon has just been dropped. There are several obvious reasons for this. For example, after a nuclear device is detonated a huge cloud is produced in a relatively short time and the associated fallout from the weapon could be hazardous to the bomber and its crew. Both of these facts are used in limiting the number of targets considered for inclusion in a route once a target has been selected for a route.

There may be other aspects of bomber operations involving the deployment of strategic gravity weapons that USTRATCOM could use. However, most of this information is sensitive and is, therefore, outside the scope of this research.

Maximum Distance

In order to develop a route generator based on distance, the maximum distance a bomber can fly is required. This distance is dependent on many factors. Such things as altitude, the speed at which an aircraft is operating, and the size of the bomb load affect how far a bomber can fly. Much of this information is classified, as are many aspects of bomber operations associated with nuclear weapons delivery. For this reason, some assumptions have to be made. First, the maximum flying distance for the generalized bomber carrying eight gravity weapons is 10,000 nautical miles with one aerial refueling. Next, it is assumed that all the bombers begin their routes somewhere in the United States and fly over the Artic Circle to get to the target country. The distance from the U.S. to the target country is subtracted from the maximum flying distance. This distance was found to be approximately 3800 nautical miles (using AMC's spreadsheet for determining great circle distances). The remaining distance, 6200 nautical miles, is used to determine feasible routes once at the target country. The final assumption is that after the bomber has completed its route and departed the target country, refueling is available for the trip to the recovery base or back to the U.S.

It may appear that the maximum distance was derived somewhat arbitrarily. In reality, a bomber's flying range is limited more by crew constraints than by fuel constraints. However, what is important is that the distance selected for use in the route generator be small enough to remove undesirable routes. Undesirable routes were

described in Chapter 1 as those that allow a bomber to strike targets on one coast, then strike targets on the opposite coast with a return to strike more targets on the initial coast. It is this crisscrossing of the country that makes routes undesirable. It makes more sense to have bomber routes confined to one coast or the other. What is required then is an approach that prevents this crisscrossing. If the maximum distance is not accomplishing this, then this distance can be reduced until it does. It would be ideal if there was a hard maximum distance and this distance was small enough to prevent crisscrossing.

Number of Targets per Route

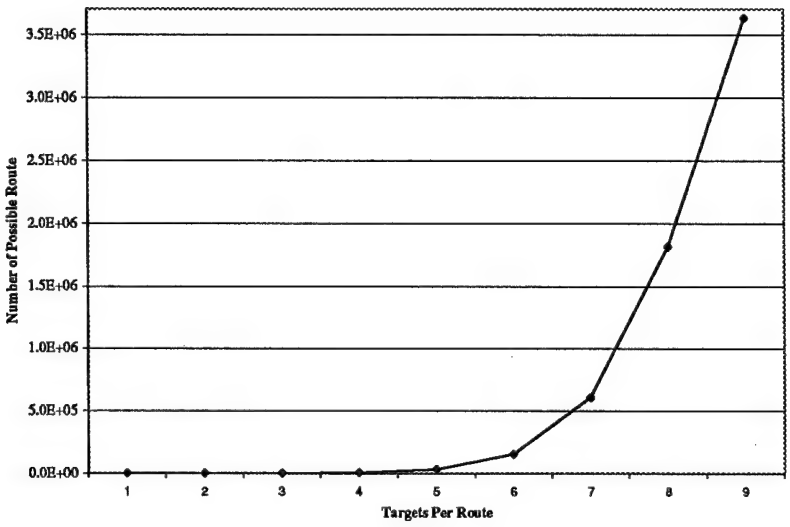


Figure 2-1. Number of Possible Routes for 10 Targets Compared to Targets Per Route

The number of targets to include per route also is needed before a procedure to generate routes can be used. This number is important because it affects the number of possible routes just as much as the total number of targets. If there are just 10 targets and a bomber can strike eight targets per sortie, the number of possible routes is approximately 1.8 million. If the number of targets per sortie is increased to nine, the

number of possible routes increases to 3.6 million. Figure 2-1 shows the nonlinear growth in the number of possible routes when just two targets are added to the route. Fortunately, bombers have a definite limit on the number of bombs they can carry. As previously mentioned, routes are generated based on a load of eight bombs. The obvious assumption is that each target gets only one bomb and all bombs are used. Therefore, all routes have eight targets.

The kind of growth in the number of possible routes seen in Figure 2-1 is what makes generating routes so difficult. By including just one more target on a route, the number of possible routes increases significantly. A similar phenomenon is observed when the total number of targets is increased by one. This means the method selected for creating routes must be capable of huge reductions in the number of routes considered to be effective.

Distance Matrix Density

Up to this point targets have been considered reachable from every other target. The distances between all pairs of targets is stored in matrix form. Matrix density refers to the number of non-zero entries in the matrix. A matrix is said to be dense if a high percentage of the entries are non-zero. On the other hand, if a matrix contains mostly zero values, it is said to be sparse. An example of a distance matrix used in this research is shown in Table 2-1. It easy to see that this matrix is symmetric with the diagonal values equal to zero. The diagonals are zero because the distance between any target and itself is zero. These are the only zero values in the matrix because of the assumption that all targets are reachable from each other. This is a dense matrix. It is obvious that if this were really the case, a route generator would produce the maximum possible number of

feasible routes. This is clearly not the case. Some targets are not reachable from each other. There are several reasons for this. Geography and target defenses are two things that may prohibit travel between two targets. If this were the case, then a zero could be placed in the distance matrix between these two targets. This increases the sparsity of the distance matrix. The sparser the distance matrix becomes, the fewer the number of possible routes that could be produced from that matrix. In this research, the worst case, reachability between all targets, is assumed.

Generating the Distance Matrices

As already mentioned, the distance between pairs of targets are needed. These distances were found by using an equation from an AMC excel spreadsheet for finding great circle distances. In order to speed the process, a macro was created that applied the equation to all possible target combinations, thus creating the distance matrix shown in Table 2-1.

Table 2-1. Distance Matrix for 10 Target Problem (Trial 10)

Target	1	2	3	4	5	6	7	8	9	10
1	0	1250	1991	273	2121	377	293	949	1478	2424
2	1250	0	1346	991	1749	1075	1391	1283	261	1596
3	1991	1346	0	1866	497	1623	2262	1223	1456	474
4	273	991	1866	0	2060	387	423	975	1211	2273
5	2121	1749	497	2060	0	1745	2411	1216	1897	781
6	377	1075	1623	387	1745	0	666	596	1329	2066
7	293	1391	2262	423	2411	666	0	1242	1591	2683
8	949	1283	1223	975	1216	596	1242	0	1538	1695
9	1478	261	1456	1211	1897	1329	1591	1538	0	1633
10	2424	1596	474	2273	781	2066	2683	1695	1633	0

For all sample problems, targets were generated in the North and East Hemispheres only. This simplifies the distance calculations. If targets had been in different hemispheres, latitude and longitude values would have to converted to + and - values to perform the distance calculations. The AMC equation used:

Y_1 = Latitude of 1st target in radians
X_1 = Longitude of 1st target in radians
Y_2 = Latitude of 2d target in radians
X_2 = Longitude of 2d target in radians

$$\text{Distance} = 60*\text{ArcCos}(\text{Sin}(Y_1)*\text{Sin}(Y_2) + \text{Cos}(Y_1)*\text{Cos}(Y_2)*\text{Cos}(X_2 - X_1))*(180/\Pi)$$

The Visual Basic in Excel does not have an ArcCos function. Excel does give an algebraic function that approximates ArcCos, and it is used in the macro. This means that some distances may be slightly different from those found using the ArcCos representation.

To get a good feel for the size of the 200 target problem, it took the distance matrix building macro over two hours to generate the 40,000 distances required to create the distance matrix.

3. Literature Review

Introduction

This chapter begins by describing the bomber routing problem in terms of a traditional mathematical programming problem. It continues by examining the difficulties with solving this problem. The last section explains the basis for the alternate approach used to generate routes without solving the mathematical problem.

The Vehicle Routing Problem

The problem of finding routes for the USSTRATCOM bombers falls into a class of mathematical programming models called the Vehicle Routing Problem (VRP). The simplest VRP can be thought of as a single vehicle with a limited capacity that must deliver a portion of its capacity to customers at various locations. In this case, the number of bombs carried by each bomber is the vehicles' capacity. The vehicle starts its route at a central location or depot (like a warehouse, factory or air base), and returns after making all of its deliveries. Actually, the VRP is just an expansion of a more common mathematical programming problem called the traveling salesman problem (TSP). In the classic TSP, one salesman is required to visit a specified number of cities in the minimum distance. In fact, the VRP is just a TSP with multiple salesmen. Obviously, there is more than one bomber in this problem and that is why it is a VRP. This problem is different from most VRPs because the vehicles are not required to return to the depot. Due to the long distances the bombers fly, they are not required to return to or recover at the same location as the one where they began. If the locations where the bombers recover are considered targets, the problem can still be formulated as a VRP.

Before entering the target area, bombers usually congregate in refueling areas north of the target country. The reason for this has already been explained. If these refueling areas are located far enough apart so that each could be viewed as a unique starting point, then the bomber routing problem could be formulated as a VRP with multiple depots.

Both VRP and TSP problems can be formulated as mathematical programming problems and are considered to be special types of integer programming problems called networks. Network problems consist of nodes (targets or customers) and arcs connecting these nodes. The arcs have a cost associated with using them. In the bomber routing problem the cost is the distance between the two nodes (targets). The objective of most VRPs is to find the best (shortest or cheapest) route or set of routes that connects all the nodes.

The problem with the VRP and the TSP is that they fall into the category of integer programming problems that are hard to solve. In fact, the VRP has been shown to be NP-hard by many researchers. Even with current computer technology, larger forms of this problem remain unsolved. Heuristics exist that provide solutions, but not necessarily the optimal solution. Most heuristics are designed to find good (short) routes. For these reasons, VRPs and techniques used to attempt to solve them have received a great deal of attention. The bomber routing problem is large. Assuming bombers can deliver 10 gravity bombs per sortie, if the number of targets is 50 and there are arcs that connect each target to every other target, then the number of routes to be considered in finding the optimal solution is on the order of 10^{16}. In this thesis, the goal is not to find the optimal set of routes; rather, the goal is to find a feasible set of routes. This requires searching a large number of possible routes. One aspect of the bomber routing problem

that makes it slightly easier to solve is that not all routes need to be considered. For example, the distance between some targets may be extremely long and exceed bomber range. These routes can be summarily dismissed. Another reason for not considering a route is enemy air defenses may make some routes infeasible or too costly. Due to the large number of possible routes, the method for generating routes from this research may require partitioning of the target set into several smaller sub-sets. At first thought, an easy way to divide the targets might be to only consider routes between targets within a small rectangle formed by the intersection of latitude and longitude lines. However, because modern aircraft are capable of flying such great distances, this may prevent the generation of desirable routes. Another way to divide the targets would be to select a sub-set at random and generate feasible routes between targets in the sub-set and repeat this process a given number of times.

VRP Formulation

The USSTRATCOM bomber routing problem as described to this point can be formulated as a basic VRP. In actuality, there are many other ways to formulate this particular routing problem. The problem's formulation depends on the objective and the constraints. The following formulation is based on the description of the problem thus far. As can been seen from the following formulation, taken from Fisher and Jaikumar [5:110], the bomber routing problem has a formulation, where the decision variables are all binary.

Constants

K = number of bombers.
n = number of targets.
b_k = capacity of bomber k.

c_{ij} = cost of travel between targets i and j.

Variables

$y_{ik} = 1$ if target i is struck with a bomb from bomber k, 0 otherwise.
$x_{ijk} = 1$ if bomber k travels directly from target i to target j, 0 otherwise.

Formulation

$$\text{Min} \sum_{ijk} c_{ij} x_{ijk}$$

Subject To:

$$\sum_{i} y_{ik} \le b_k, \qquad k = 1,...,K \tag{1}$$

$$\sum_{k} y_{0k} = K, \tag{2}$$

$$\sum_{k} y_{ik} = 1, \qquad i = 1,...,n \tag{3}$$

$$y_{ik} = 0 \text{ or } 1 \qquad i = 0,...,n \qquad k = 1,...,K \tag{4}$$

$$\sum_{i} x_{ijk} = y_{jk}, \qquad j = 0,...,n \qquad k = 1,...,K \tag{5}$$

$$\sum_{j} x_{ijk} = y_{ik}, \qquad i = 0,...,n \qquad k = 1,...,K \tag{6}$$

$$\sum_{ij \in S x S} x_{ijk} \le |S| - 1, \qquad S \subseteq \{1,...,n\} \quad 2 \le |S| \le n - 1 \tag{7}$$

$$x_{ijk} = 0 \text{ or } 1, \qquad i = 0,..,n \qquad j = 0,...,n \qquad k = 1,...,K \tag{8}$$

The depot is the first element of the set of targets and corresponds to $i = 0$. Constraint (1) ensures the capacity of each bomber is not exceeded. Constraints (2) and (3) ensure that each route begins and ends at the depot and that every target is serviced. These are the constraints that have to be modified for the bomber problem. As already mentioned, the bomber formulation includes recovery bases as targets. Then there are

equations that require each route to end at these recovery bases. In addition to this change to Constraint (3), it is likely that there will be more targets than bombers. If this is the case, then not all targets have to be visited and the constraints in (3) will be replaced by constraints that force the routes to end at a recovery base. Constraints (5) through (8) define a TSP over the customers assigned to bomber k. In Constraint (7), S is a subset of the n targets and $|S|$ represents the cardinality of S. Cardinality refers to the number of elements in set S.

The objective is to minimize the distance traveled. This objective can be easily changed with distance converted to a constraint. There may be other more important objectives that need to be satisfied by the bomber routes. Examples are to maximize the probability of survival or to maximize the value of striking target i with a gravity weapon. If this were done, the new objective function might be:

$$\text{Max} \sum_{ijk} p_{ij} x_{ijk}$$

where p_{ij} is the probability of surviving the flight from target i to j. The distance objective would now be a constraint:

$$\sum_{ij} c_{ij} x_{ijk} \leq D \quad \forall K$$

where D is the maximum distance the bomber is able to fly.

Heuristic For Solving VRPs

It has already been mentioned that many researchers have shown that the VRP is difficult to solve, and for large VRPs, no known algorithm exists that guarantees finding an optimal solution. The TSP for just 25 cities has just over 10^{25} possible solutions. Therefore, many have attempted to use heuristics to solve VRPs. Several modern

heuristics like Genetic Algorithms, Simulated Annealing, and Tabu Search (TS) have been used on many different types of VRPs and have had a great deal of success finding good solutions. For example, Renaud et al [12] showed that his TS heuristic found the best known solutions for 20 of 23 benchmark Multi-Depot VRPs.

In general these heuristics start from some initial (usually feasible) solution and attempt to move to a better solution. The methods continue until they have reached some established stopping criteria. TS, for example, as developed by Glover, Taillard and de Werra [3], is an iterative method that moves from one solution to another in the neighborhood of the last solution. The idea is to move in a direction that improves on the previous solution in terms of some objective function, in this case, the shortest route. Once a good solution has been found, TS identifies this solution as tabu and does not allow a return to that or similar solutions for a given number of iterations. When a solution has been identified as tabu, it is generally placed on a list with other tabu solutions until the specified number of iterations has passed. The typical TS heuristic is stopped when there is no improvement in the solution after a given number of iterations. Obviously, this method does not generate all the feasible routes. It does generate a large number of feasible routes, and, most importantly, it generates short routes. It is desirable to have the shortest known routes in the collection of feasible routes. The most common TS heuristic would have to be modified to produce a list of all feasible routes since TS usually only outputs the best solution found. The problem with these types of heuristics is the fact that they can be complex and that the bomber route problem is embedded in an optimization problem. This optimization problem is solved by another method, namely WAM. The bomber routing problem is better thought of as a problem requiring

generation of many feasible routes. A specific heuristic for this problem should be easier to implement and do a better job of generating feasible routes than any of the existing heuristics used to solve the VRP.

Preprocessing and Partitioning

In order to speed the process of generating feasible routes, the target database may be partitioned into smaller sets. This partitioning may be done outside of the route generation process. Processing data prior to manipulating it is a common practice with large integer programs and is referred to as preprocessing [9]. If the refueling areas are considered to be the sources and the targets are considered to be the sinks, the bomber routing problem can be preprocessed by formulating an assignment problem. This model assigns targets to particular refueling areas. This kind of assignment has been done with VRP and multiple depots by Fisher and Jaikumar [5]. A formulation like this for the bomber problem should attempt to minimize the sum of the distances flown. The constraints would ensure that each target is assigned to only one refueling area.

The idea of preprocessing the VRP is not new. An idea suggested by Taillard [14] is that targets (nodes, customers) near each other would be served by the same depot. This idea makes sense when considering large problems where each depot services a large area. If a target is located a long distance from one refueling area and near another, it may not make any sense to include it in routes from the most distant refueling area. Targets located similar distances from different refueling areas present a problem. One way around this problem is to arbitrarily assign these targets to the refueling area with the fewest number of assigned targets.

Another way to preprocess the target database is to simply divide the targets into groups based on their location. Squares or rectangles using lines of longitude and latitude could be used to separate the bomber route generation problem into several smaller sub-problems. The draw back to this method is it eliminates routes between targets in different geographic regions and this may not be desired.

Implicit Enumeration

Implicit enumeration is used on problems that involve a large number of possible solutions like the bomber route generation problem. It reduces the number of solutions considered by removing groups of possible target combinations. Once a target or group of targets are set in an order on a route, other targets may be removed from consideration. One reason for elimination of a target is that the distance to that target from the previous target makes the route too long. If a target is eliminated from consideration, all combinations that include that target in that position on a route are eliminated. This procedure is called implicit enumeration because solutions are enumerated without being generated [9].

4. Methodology

Introduction

A majority of Chapter 4 is devoted to developing the approaches that are used to generate bomber routes. Four approaches are presented. The first uses only the distance between targets and is called the distance generator. The next two were created from the concept that most bomber routes begin in the northern part of the target country and move southward. The first of these two approaches allows the flight path to move only to the south from each target and is called the north south walk. The next approach allows the direction of flight to change, either slightly to the north or the south at each target. It is called the north south walk with backtracking. The last approach, known as the true backtracking generator, allows the direction of flight to vary. It also prevents the bomber from turning around after striking a target and flying over the area just bombed. In the remaining sections of this chapter, unless stated otherwise, it is assumed that a complete route includes eight targets. The last section of the Chapter explains the methods employed to create sample problems for testing the performance of the generators.

Distance Generator

The route generator based on distance is quite simple. The generator begins at a given target and then selects the next target from the pool of those remaining. The distances between all pairs of targets are generated prior to running the distance generator. Table 2-1 gave an example of a distance matrix. The method for calculating the distance between two targets involves several mathematical operations. A distance is

used numerous times so recalculating it several times is not efficient. To save time the distance matrix is created prior to executing the route generator.

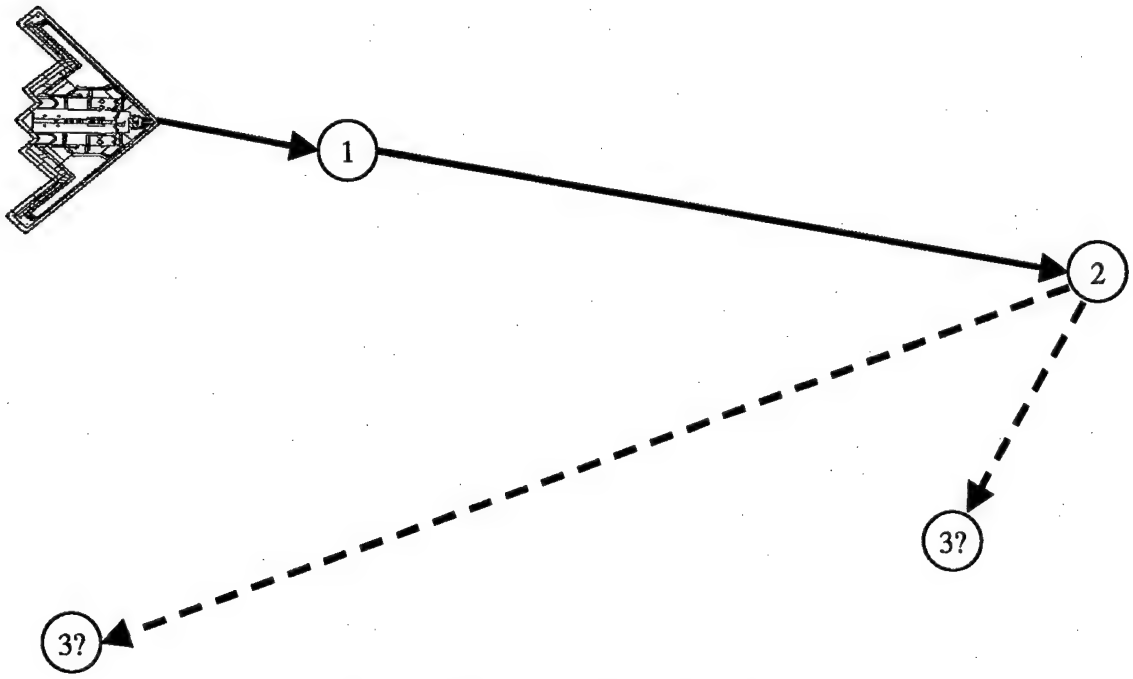


Figure 4-1. Distance Generator

Since the distance matrix is symmetric, the distance generator only needs to use half of this matrix. The length of a route after a target is added is compared to the maximum distance previously discussed. If it is greater than the maximum distance, the target is discarded and another target is selected from those remaining. If the resulting route length is not greater than the maximum distance, this second target is stored and the generator continues the route building process. For example, in Figure 4-1 the distance between target 1 and target 2 is added to the distance between target 2 and target 3. If this distance is greater than the maximum allowable distance, the target is removed from consideration. If the distance is not greater than the maximum allowable distance that target is added to the route and the process continues. The generator continues to add and discard targets in this fashion until a complete route is created. Once a route is generated,

the approach discards the last target added to the route and starts cycling through the pool of remaining targets trying to build more routes. If there are no targets that create a new feasible route or the approach finishes cycling through the remaining targets, the approach drops the last two targets and continues trying to create new feasible routes. When a target is discarded, all possible permutations involving the target in the current position on the route are also eliminated from consideration. This is where the most significant reduction in the number of routes considered occurs with an associated reduction in processing time. Thus, the generator uses implicit enumeration to reduce the number of potential routes evaluated.

It is difficult to determine the exact number of mathematical operations performed by the generator because of the way the generator discards possible routes. The implicit enumeration method used by the distance generator does not create all possible permutations of the targets, and just where and when a possible route or groups of possible routes are eliminated is not easy to determine. For this reason, to compare the different route generators, the number of operations required to create a single route is determined. The distance generator uses only addition and comparison operators. For all the generators, there are comparisons at each target assignment to ensure the target being considered is not already assigned to the route. Obviously, this comparison is not required at the first target. Then, the number of comparisons at each step increases by one as each target is added to a route. Since all the generators require these comparisons they are left out of the total. Again, except for selection of the first target, there is one distance comparison at each iteration for a total of seven. Now, except for the first two targets, there are addition operations required to find the total distance flown. At each

step the distance between the current target and next target is added to the total of all previous distances flown. Thus, the total number of addition operations for the distance generator is seven. For comparison to the other generators, the number of total operations for the distance generator is 14.

North South Walk

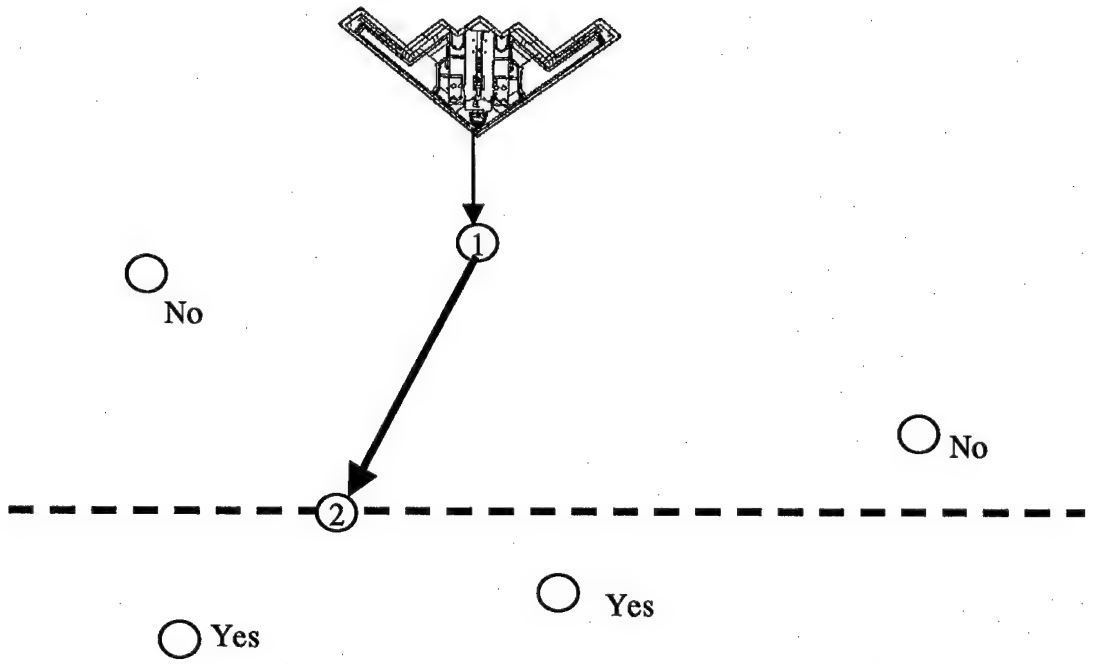


Figure 4-2. Target Exclusion for the North South Walk

Another way to generate routes is based on two simple assumptions mentioned in Chapter 2. The first assumption is that most, if not all, routes begin by flying over the Artic Circle and the second assumption is that bombers do not turn around and fly over any area just attacked. This generator based on these two assumptions is called the north south walk. For this generator, target locations in latitude and longitude are transformed to an x and y coordinate system. The approach begins by selecting any target from the target base. Then, it only allows targets to the south of the first target to be considered.

Essentially, this draws a latitude line through the target and removes all targets lying north of the line from further consideration as is shown in Figure 4-2. Once a target to the south of the first is located, it is added to the route. The same process is repeated at each target. Given the following definitions and the standard equation for a line, $Y = MX + B$, where

Y_1 = Latitude of the current target
X_1 = Longitude of the current target
Y_2 = Latitude of the next target
X_2 = Longitude of the next target

then the equation of the line used to determine target acceptability at each step is

$$Y = Y_1$$

At each step the generator checks to see if $Y_2 \leq Y_1$.

If there are no targets to the south of the current target, the generator discards this target and tries another. If there is a target to the south of the current target, the generator adds the current target to the route. Again, it continues to do this until a route of eight is constructed, and like the distance generator, it then attempts to dismantle and rebuild the route. Also, like the distance generator, it uses implicit enumeration to reduce the number of routes considered.

The number of operations performed by this generator is easily determined. The generator only compares the latitude of the current target to the latitude of the previous target. Therefore, the north south walk generator only requires seven successful comparisons to form a route.

North South Walk with Backtracking

The north south walk applies a fairly strict constraint on all the routes it generates. It does not let the bomber fly north. It has been assumed that the bombers do not turn around and fly over an area just bombed. However, this does not mean that the bomber does not turn after a bomb is dropped. To permit the bomber to attack targets north of the current target, a route backtracking capability is included in this generator. To allow the bomber some backtracking capability, instead of drawing a latitude line through the target, the north south walk with backtracking generator draws two lines. These lines, as shown in Figure 4-3 form 45 degree angles with the longitude line that runs through the target. These are easy to find.

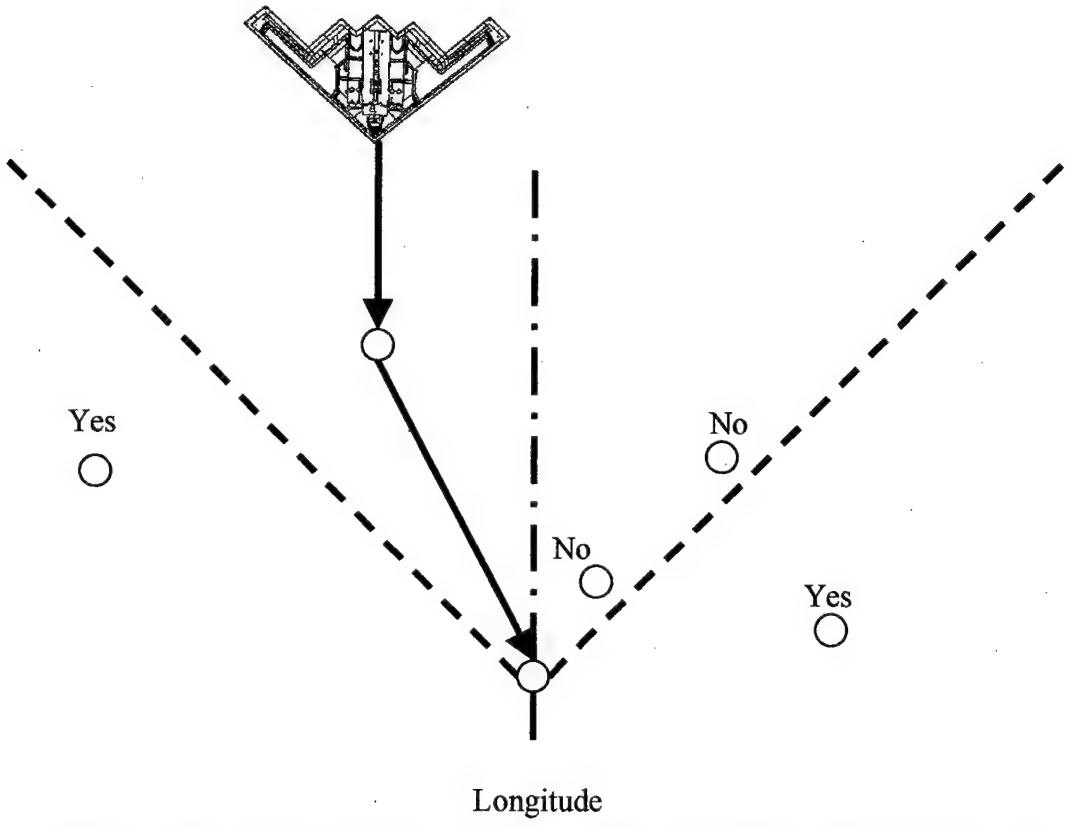


Figure 4-3. Target Exclusion for the North South Walk with Backtracking Generator

The slopes are simply +1 or -1. Therefore the equations of the lines in terms of the current target are:

$$Y = 1*(X - X_1) + Y_1 \text{ and } Y = -1*(X - X_1) + Y_1$$

The next target lies below these lines if:

$$Y_2 \leq X_2 + Y_1 - X_1 \tag{9}$$

or

$$Y_2 \leq -X_2 + Y_1 + X_1 \tag{10}$$

The generator skips targets that do not meet these conditions. To accomplish this, two cases are created so that the next target only needs to be evaluated against one line or the other. The most logical way to do this is to first determine which side of the longitude line running through the current target the next target is on. If the next target is west of the current target it is evaluated using equation (10), and if the next target is east it is evaluated using equation (9). Then, the psuedo code for this part of the north south walk with backtracking generator is:

```
If X2 ≥ X1 Then
        If Y2 ≥ X2 + Y1 - X1 Then
                GoTo next target
        End if
Else if X2 < X1
        If Y2 ≥ -X2 + Y1 + X1 Then
                GoTo next target
        End if
End if
```

To find the number operations performed to create a route with the north south walk with backtracking generator is more difficult than in the previous two generators. Like the previous two generators, everything starts at the second target. The first thing the generator does is determine where the target under consideration lies in relationship

to the current target in terms of longitude. This requires one comparison. Then it checks to see if the next target lies above the line that forms a 45 degree angle with the longitude line of the current target. To do this requires, one comparison, one addition, and one subtraction or one comparison and two additions depending on the target's location. Then, for each target selected there are three more operations. This yields four operations every time a target is added to a route and the total number of successful comparisons for a complete route generated by the north south walk with backtracking is 28.

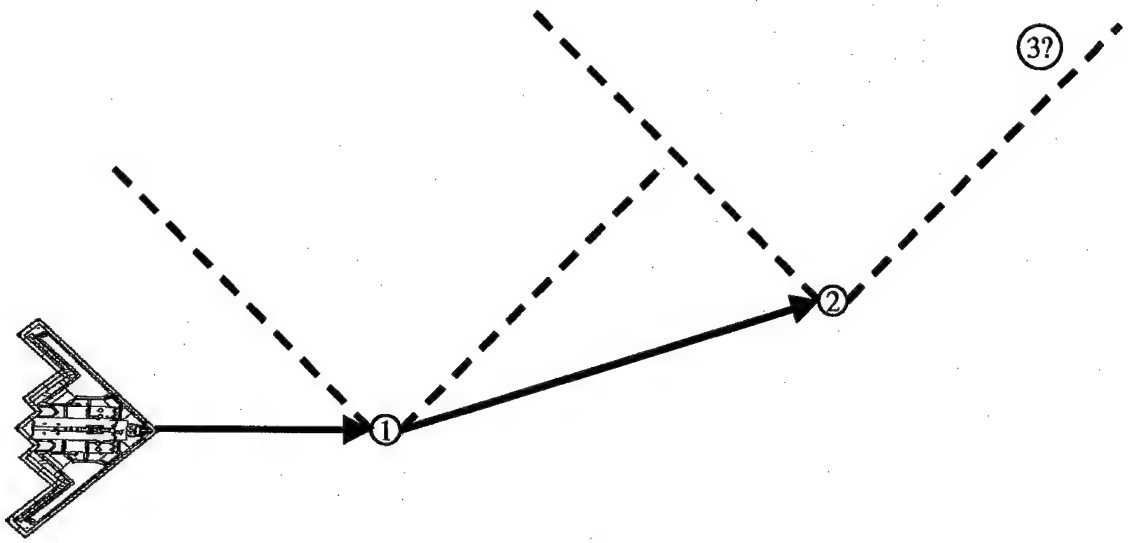


Figure 4-4. A Desirable Route Excluded by the North South Walk with Backtracking

From Figure 4-3, it is easy to see that this approach gives a good set of routes that move from north to south and allows route backtracking. In theory, a route moving slowly to the north can be created. Regardless of which target was selected prior to the current target, the same two lines are drawn for that target every time. This presents a problem. Targets that are in front of the bomber are eliminated from consideration because the orientation of the two lines does not change with the direction of flight. Thus, this method of backtracking is not ideal. In Figure 4-4, routes that include target 3

after striking targets 1 and 2 should be feasible, but target 3 is always eliminated from consideration after this generator assigns target 2 to a route. It seems the best approach is to allow the 45 degree lines to move in relation to the bombers flight path.

True Backtracking Generator

In order to create the generator eluded to in the previous paragraph, an efficient method for finding the slope of the 45 degree lines in relation to the current flight path needs to be developed. Before this can be done, a method for expressing the slope of the current flight path and these 45 degree lines is required. The true backtracking generator functions by moving the intercept of an x and y coordinate system to the location of the current target. Figure 4-5 displays this x, y coordinate system with the intercept of the two axes occurring at the location of the current target. The following equations are equivalent expressions for the slope of the current flight path in terms of the tangent of the angle formed by positive x axis and the flight path. When the intercept of the x, y

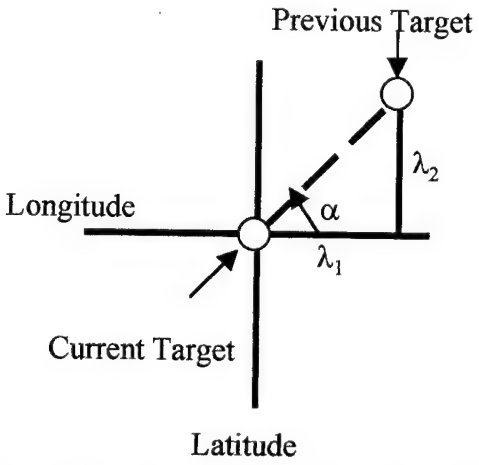


Figure 4-5. Slope of the Current Flight Path and Tangent Diagram

coordinate system is moved to the current target's location, the slope of the line connecting the current target with the previous target is equal to the tangent of the angle formed by the line connecting these two targets and the x axis.

Let Y_0 = Latitude of previous target
X_0 = Longitude of previous target

Then, the slope of the current flight path, M_C, can be expressed in the following forms:

$$M_C = (Y_1 - Y_0)/(X_1 - X_0) = \frac{\sqrt{(Y_1 - Y_0)^2}}{\sqrt{(X_1 - X_0)^2}} = \frac{Length(\lambda_2)}{Length(\lambda_1)} = \tan(\alpha) \qquad (11)$$

Clearly, there are two ways to represent the slope of the line connecting the current target with the previous one. This generator uses $(Y_1 - Y_0)/(X_1 - X_0)$ because the computer can perform the calculations in this expression faster than it can calculate the tangent. Then to find the slope of a line b degrees from the current flight path:

$$M_{C+b} = \tan(\alpha + b) = [\tan(\alpha) + \tan(b)]/[1-\tan(\alpha)\tan(b)] \qquad (12)$$

Angle α is the flight path angle to the current target from the previous target. Then, to find the slope of the line 45 degrees greater than the approach angle, angle b is +45 degrees and the $\tan(b) = 1$. Then equation (12) is:

$$M_{C+b} = (M_C + 1)/(1 - M_C) \qquad (13)$$

To find the slope of the line 45 degrees less than the current flight path, angle b is -45 degrees and the $\tan(b) = -1$. Now equation (12) is:

$$M_{C-b} = \tan(\alpha + b) = (M_C - 1)/(1 + M_C) \qquad (14)$$

The equation of the two associated lines are developed like the equations for the north south walk with backtracking. They are given by:

$$Y = M_{C+b} * (X - X_1) + Y_1 \qquad (15)$$

and

$$Y = M_{C\text{-}b} * (X - X_1) + Y_1 \quad (16)$$

As in the north south walk with backtracking, care must taken when determining where the next target lies and which line to compare it to. For this generator, the slope of the current flight path and the latitude or longitude of the previous, current and next targets are used to divide the problem into eight cases. The latitude or longitude of the current target combined with the slope is used to determine the main direction of flight, north, south, east or west. The latitude and longitude of the next target are used to determine which quadrant it is in and which line it should be compared to, the line with slope M_{C+b} or the line with slope M_{C-b}.

The Route Building Method of the True Backtracking Generator

Step 1. Assign the first and second targets to a route.

Step 2. Determine the slope of the line connecting the previous two targets. Using this slope, find the slope of the line 45 degrees greater than and 45 degrees less than this line.

Step 3. Determine the direction of motion of the route at the current point based on the slope of the current flight path and the latitude or longitude of the current target. The choice of latitude or longitude depends on the value of the slope of the current flight path. If the slope is between -1 and 1, the longitude of the last two targets is used. If the slope is greater than 1 or less than -1, then the latitude of the last two targets is used. This is done to aid in deciding which line to evaluate the next target against.

Step 4. Determine the location of the target being considered for addition to the route in relation to the current target. Like the previous step, this step is done to determine which line the next target should be evaluated against and is combined with the results of the previous Steps to make this determination. If the slope is between -1 and 1, the latitude of the current target and next target are used to determine if the target is located to the north or south of the current target. If the slope is greater than 1 or less than -1 the longitude is used to determine if the target is east or west of the current target.

Step 5. Evaluate the target being considered against the appropriate line formulated in equations (15) and (16). If the target is accepted, add it to the route

and return to Step 2. If it is not accepted, discard the target, select another for consideration, and return to Step 3.

The case where the slope of the current flight path is between 0 and 1 and the longitude of the current target is less than the longitude of the previous target is seen in Figure 4-6. If the latitude of the next target is greater than the latitude of the current target, it must lie below the M_{C+b} line to be removed from consideration. If the latitude of the next target is less than the latitude of the current target it must lie above the line with slope M_{C-b} to be excluded from consideration.

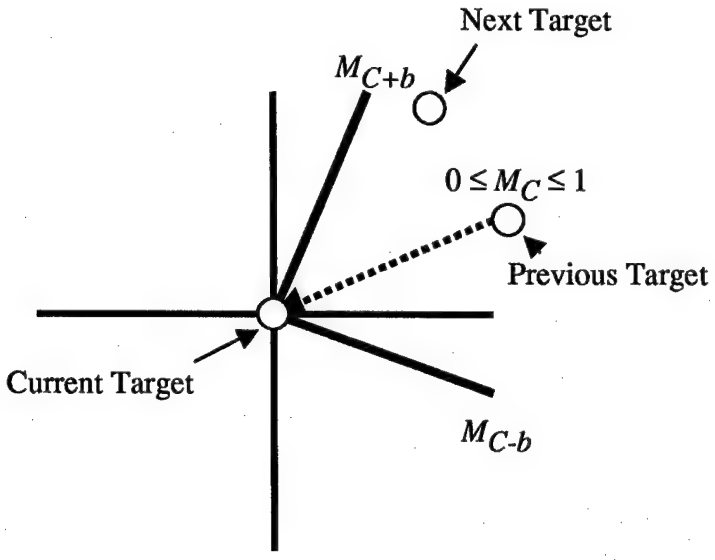


Figure 4-6. Method for Target Exclusion Using the True Backtracking Generator ($0 \leq M_C \leq 1$)

The psuedo code for the previous example is:

```
If 0 ≤ MC and MC ≤ 1 Then
  If X1 ≤ X0 Then
    If Y2 ≥ Y1 then
      If Y2 ≤ MC+b*(X2 - X1 ) + Y1 Then
            Discard The Target at (X2, Y2)
      Else Add To Route and Continue
```

```
        End if
      End if
    End if
End if
```

The other cases are similar to this case and are separated by slopes of the current flight path in these ranges: between 0 and -1, greater than 1, and less than -1.

It is clear from the previous explanation that this generator is far more complex and requires more operations. At each step, the slope of the current flight path and the slope for each 45 degree line from the current flight path must be found. For each slope calculation there are two additions or subtractions and one division, so the total is six additions/subtractions and three divisions per target excluding the first target. Then, there is one comparison to determine the direction of the current flight path and one to determine the location of the next target in relation to the current target. Once this is complete, this generator is like the north south walk with backtracking generator. It compares the next target to a line. The difference is that this generator requires a multiplication since the slope of the line is likely not equal to ± 1. Then the total for this step is one comparison, one multiplication and two additions/subtractions. Then for each target assigned there are three comparisons, eight additions/subtractions, and four multiplications/divisions, which gives a total of 105 operations to determine a route. This is a significant increase compared to the previous generators.

Limitations of the True Backtracking Generator

There are two situations that may cause problems for the true backtracking generator. The first situation is when the longitude value of the current target and the longitude value of the previous target are very close or equal. As these two values

approach equality, the denominator of the calculation to find the slope of the current flight path approaches zero. This means the value of the slope becomes very large and if the denominator equals zero the value of the slope is not defined. This situation can cause problems with the computer that could slow or even stop program execution. The second situation is when the value of slope of the current flight path is one or negative one. If this is the case, then depending on whether the value is one or negative one, either M_{C+B} or M_{C-B} is undefined, which could stop program execution. This can be avoided by having the program check for these instances at each step. Each of the situations dictates values of the slope of the lines M_{C+B} or M_{C-B} that can be used for target exclusion.

Neither of these situations appeared in the sample problems because of how the targets were generated and the relatively few number of targets used. The method for generating targets is explained in the Sample Problem section of this chapter. Therefore, the route generators do not have a check for these situations. It is possible that the real target database may have one of these problems, because real targets are not randomly generated. So, it would be necessary to incorporate checks for these special situations into the true backtracking generator if it is used with an actual target database.

Combinations

The only generator that prevents the aforementioned crisscrossing of the country is the distance generator. The distance generator can be included in any of the other generators. So the distance generator can be used in combination with any of the other generators to reduce the number of routes considered. The problem with this is it adds additional calculations and comparisons to the generators and as the size of the problem

increases, the generators' solution time increases significantly. Hopefully, the reduction in routes generated outweighs the added computational effort.

Sample Problems

Several sample problems were created to test the performance of the generators developed in this chapter. The two main types of sample problems used for testing were, eight targets with routes of four targets and 10 targets with routes of eight targets. Initially, two other problems were tried. They were 50 possible targets with five targets per route and 25 possible targets with five targets per route, but because there were only five targets per route the distance generator did not perform well. The problems were created in Microsoft Excel and the generators were coded in simple macros. Excel macros use the Visual Basic programming language, which was one of the main reasons for choosing Excel. Visual Basic can be used for complicated object oriented programming, is very easy to use, and comes with a debugging program that simplifies troubleshooting.

Targets were generated using the random number generating function RAND() provided by Excel. RAND() generates random numbers between 0 and 1 from a continuous uniform distribution. This means the likelihood of getting any number between 0 and 1 has the same probability. These random numbers are manipulated to get latitude values between 0 and 90 and longitude values between 0 and 180, depending on the sample problem. Two random numbers were generated for each target. One was used to determine the latitude and the other one was used to determine longitude. For example, to generate the eight targets for the first sample problem, RAND() was used in the following fashion:

$$\text{Latitude} = \text{RAND}()*(70-50) + 50$$

$$\text{Longitude} = \text{RAND}()*(140-100) + 100$$

Examining the latitude equation, shows that the random number is multiplied by the spread between the maximum and minimum values of latitude and then added to the minimum. This guarantees the latitude is between its minimum and maximum values of 50 degrees North and 70 degrees North. The same formula is used for determining longitude except the minimum and maximum values are 100 degrees East and 140 degrees East.

The eight possible targets with four targets per route problem was used to validate the macros developed for each of the above generators. The reason it was used to validate the generators is that for this problem there are only 1680 possible routes. The number of routes generated is less than this number, and all, or at least a large sample of the routes could be checked manually to ensure the generators were functioning properly. The selection of latitude and longitude ranges was based on current realities. The latitude was allowed to vary between 50 and 70 degrees North and the longitude varied between 100 and 140 degrees East. Only one set of eight targets was used.

The problem with 10 possible targets with eight targets per route was used to evaluate the performance of the different generators. Again, minimum and maximum values of latitude and longitude were selected as described above. However, different values were used. Latitude varies from 40 to 70 degrees North and longitude varies from 30 to 160 degrees East. Ten sets of 10 targets were generated and the following generators were applied:

Distance
North South Walk

North South Walk with Backtracking
True Backtracking Generator
True Backtracking Generator with Distance (Combination)

To evaluate these generating approaches, a large test problem is used. If the true backtracking generator and the distance generator are the best, another way to get all the routes to start in the north needs to be used. Neither of these two generators requires routes to begin in the north. As a simple means of implementing this restriction, target sorting is used. The targets are sorted in descending order from north to south and numbered in this order. In the visual basic macros, the targets are selected by FOR loops with the target number as the looping variable. The variable used for the first position on a route is X_1. This variable starts at one and is allowed to loop over all targets. However, for the larger problems, it is allowed to loop over a selected subset of the targets: the first 40. Since the targets are sorted in descending order, then the first place on a route is allowed to loop over the most northern targets. For the next place, more targets are evaluated for inclusion on a route. At some point, prior to the last few positions on a route, all the targets are available for selection on a route. Two larger sample problems are used for testing. One has 200 targets, while the other has 100. There is no real rational behind the selection of these two sample problem sizes other than the problems needed to be large enough to challenge the generators. If 200 is too large and cannot complete in a reasonable amount of time, hopefully, all the feasible routes for the 100 target problem can be generated.

The targets for this problem are generated as in the previous sample problems using the RAND() function provided by Excel. However, the values of latitude and longitude are different. Half of the targets are generated with latitude values between 45

and 65 degrees North and longitude between 30 and 60 degrees East. The other half of the targets are allowed to vary between 40 and 70 degrees North and 30 and 140 degrees East. The reason for generating targets in this manner is to test the performance of a generator when target clustering is present.

Even if a large reduction in routes generated is achieved by the true backtracking and distance generator combination, there will be a huge number of routes produced. Therefore, it was decided to only store a small fraction of these routes to verify that the generator was working properly. Not outputting all routes reduces the processing time. This needs to be taken into consideration when evaluating overall performance.

5. Results and Testing

Processing Time

The calculation of processing time for each generator is identical. For this research, the processing time is the time the generator is actually at work generating routes. In the visual basic macros, after variable declarations and initializations the program checks the current time using the NOW() or the Timer functions. This time is stored. After the last route has been generated and prior to the output of information generated by the program, another check of the time is made and this time is stored. To calculate processing time, the difference between the last time and the first time is evaluated to find the elapsed time. In the two large sample problems the NOW() function is used. Exactly how the NOW() function operates is explained later, but the reason it is used for the larger problems is because elapsed times of greater than 24 hours can be calculated. In the smaller sample problems, the Timer functions is used because of the shorter processing times. The Timer function simply returns the time in seconds since midnight.

The NOW function uses a calendar that starts on January 1, 1900 and assigns a number to any date and time that occurs after this. Depending on the date and time, the number can be rational or integer. It is this fact that permits the calculation of longer elapsed times. If the number is a rational number, the value to the left of the decimal represents days since January 1, 1900 and the number to the right the fraction of a day. For example, in the January 1, 1900 date system, the serial number 367.5 represents the date-time combination 12:00 P.M., January 1, 1901.

Distance Generator

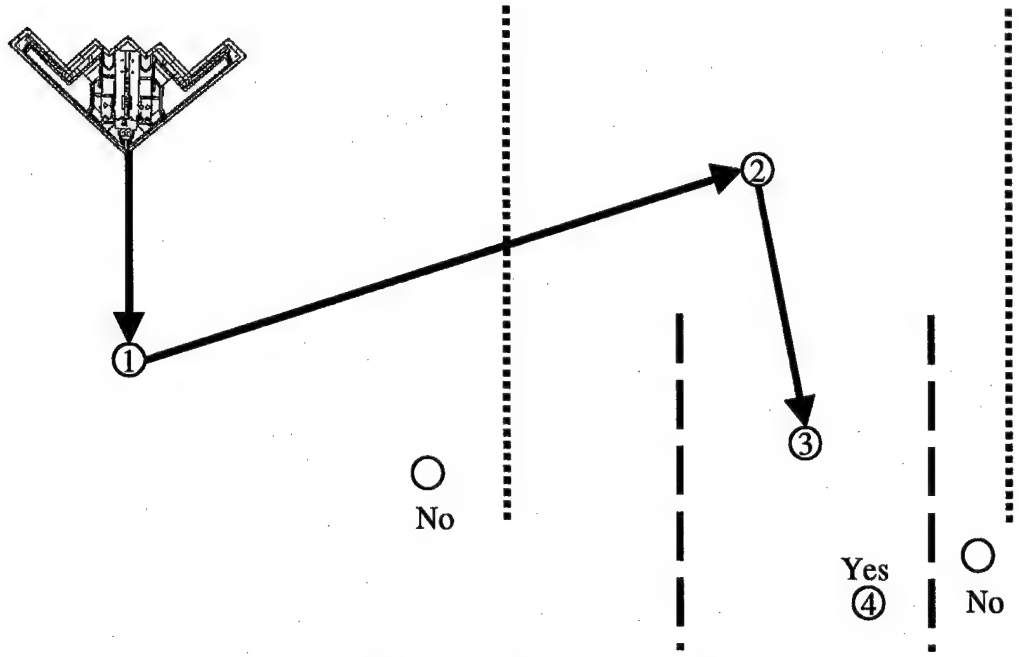


Figure 5-1. Longitude Funnel

The importance of the selection of a good maximum distance has already been briefly discussed. However, the discovery was not made until after some work with the smaller sample problems and the distance generator had been completed. Initially, in the eight target problem, the reduction in routes generated by the distance generator versus the number of possible routes was not large. Several distances were attempted, but no great improvement was made until the maximum distance was reduced a significant amount or the targets were spread over a wider area. It was felt that the reduction in distance was too large and might be removing some desirable routes. This suggested that the distance for this problem would have to be manipulated some other way or another method developed to prevent the nongeneration of desirable routes. At first, another method was proposed. If, like that north south walk generator, longitude is restricted based on previous target selection the problem may be solved. This method is referred to as the longitude funnel. As the generator begins assigning targets to a route, the

longitude of the next target is allowed to vary over a wide range. As more and more targets are added to a route, the allowable longitude range for selection of the next target is reduced. For the eight target problem with four targets per route, after the first target is selected the next target's longitude was not restricted as in Figure 5-1.

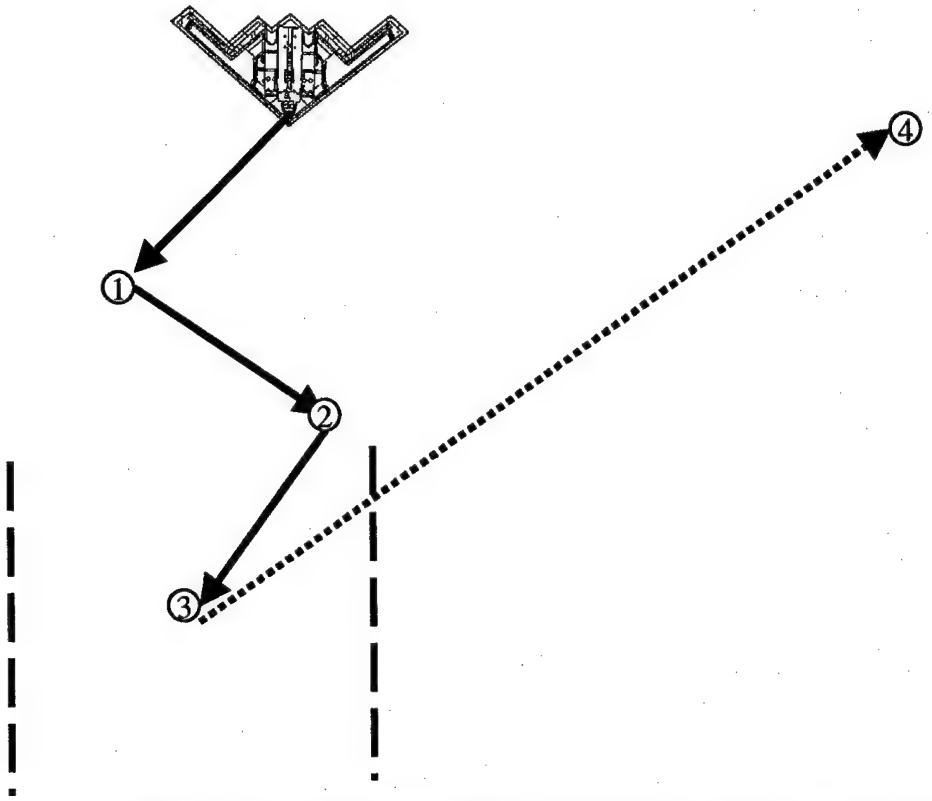


Figure 5-2. A Desirable Route Excluded by the Longitude Funnel

The third target's longitude was required to be within half of the total longitude range of the second target. The longitude range for this problem was 100 to 140 degrees, so the third target must be within plus or minus 10 degrees of the second target's longitude. Once the third target was selected, the fourth target's longitude was restricted to be within plus or minus 5 degrees of the third target's longitude. The longitude funnel did significantly reduce the number of routes generated. For the eight target problem, of 1,680 possible routes the longitude funnel returned only 220 routes compared to the

distance generator which created 860 routes. After examining the routes produced by the longitude funnel, at least one type of route with desirable characteristics was not generated. The type of route not generated was one where the beginning of the route has several targets clustered together and the last target is separated from the clustered targets as seen in Figure 5-2. This was deemed unacceptable so, it was decided to increase the number of targets per route and then reexamine the performance of the distance generator.

The performance of the distance generator on the sample problem with ten total targets and eight targets per route showed some promising improvements. The average number of routes generated for the ten trials performed was 15,285. For this ten target problem the total possible number of routes is over 1.8 million, so there was a reduction of approximately 0.8% in the number of routes generated. Compared to the eight target problem, where the reduction was near 50%, this is a substantial improvement. A similar result is expected when more targets are added to the standard route length of eight. As more targets are added to the routes, the average route distance increases. If the maximum distance is held constant, then because the average distance of a route increases, the distance generator's performance should improve for routes with more targets. Remember that the number of possible routes also increases when route length increases. Therefore, the conclusion was made that the distance generator should reduce the number of routes considered and created, but by itself cannot produce the kind of reduction needed to make the problem of generating routes a manageable one.

Figure 5-3 is a map of the targets for two trials of the ten target problem. The distance generator did not perform well on trial two. From the figure it is easy to see that

the targets in trial two are clustered closer together than in trial four. Obviously, the clustered targets have shorter distances between them. This allows the formation of more routes. The distance generator performed the best on the targets in trial four.

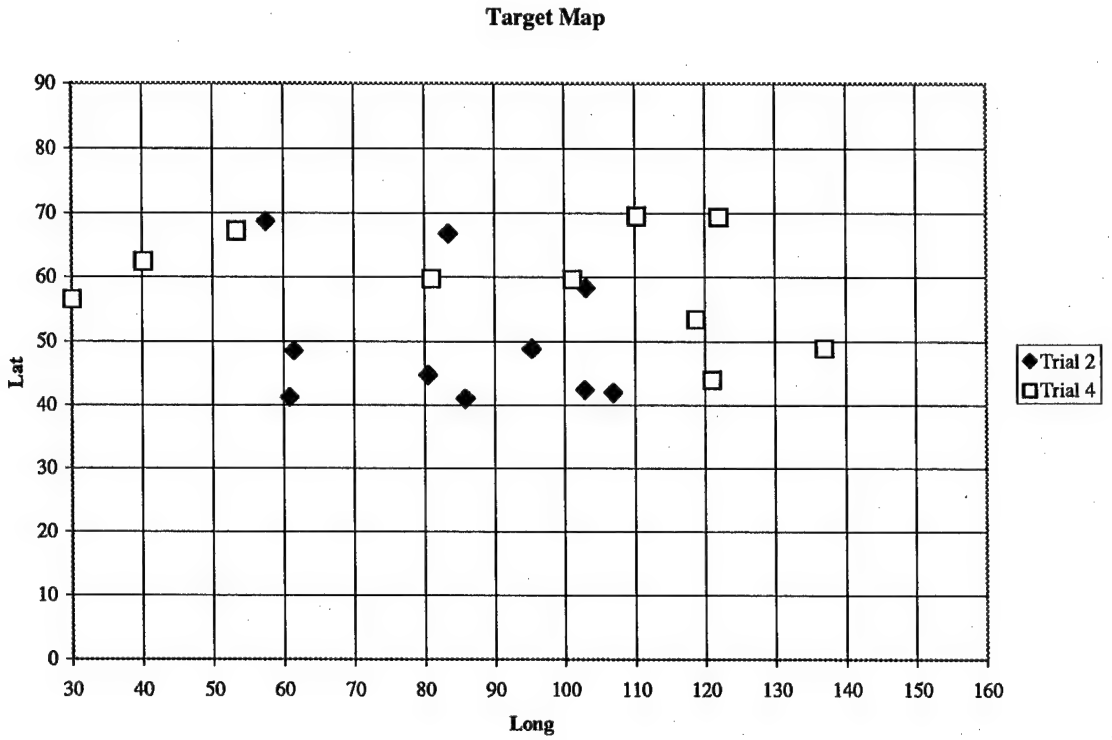


Figure 5-3. Target Map for Trials 2 and 4 of the Ten Target Problem

North South Walk

While running the North South Walk generator on the ten target problem, an interesting discovery was made. All ten runs return the exact same number of routes, 45. It was soon realized that 45 is the number of possible combinations when 10 numbers are taken in groups of eight. Statistically, combinations are different than permutations. In combinations, the order of the numbers does not matter. For example, the combination 1-2-3-4-5-6-7-8 is the same as 1-2-3-4-5-6-8-7; however, they are different permutations. A permutation is used when calculating the number of possible routes because order

matters. The number of permutations of n things taken r at time is given by the formula n!/(n-r)! The number of combinations are found in a similar way except the denominator is multiplied by r!, which gives n!/[(n-r)!*r!].

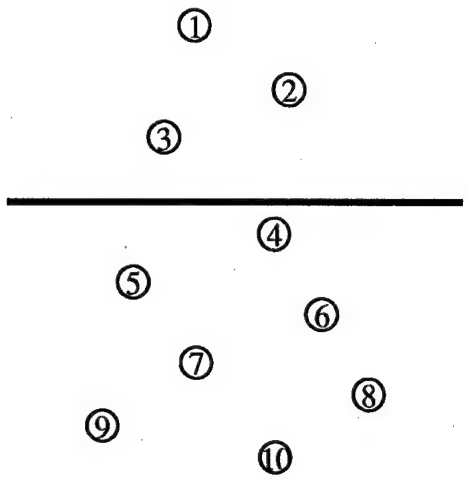


Figure 5-4. Diagram of Results From the North South Walk

Figure 5-4 shows that in order to make a route of eight targets while continuing to move south, the only feasible starting targets are 1, 2 and 3. If a route is started at target 4 or a more southerly target, there is no way to make a route with eight targets. Obviously, there are only a few ways to select the targets that will form routes that continue to move in a southerly direction. It turns out the order of selection does not matter. If the route begins at target one, there are only nine targets remaining to fill seven spots on the route. If a route begins at target 2, then there are only eight targets remaining to fill seven spots and if a route starts at target 3, there are only seven targets remaining to fill seven spots. So, the number of routes generated is equal to the number of combinations:

$$\frac{9!}{(9-7)!*7!}+\frac{8!}{(8-7)!*7!}+\frac{7!}{(7-7)!*7!}=36+8+1=45$$

Then the north south walk does not produce enough routes with desirable characteristics and this reduces the usefulness of the north south walk's method of finding routes.

North South Walk with Backtracking

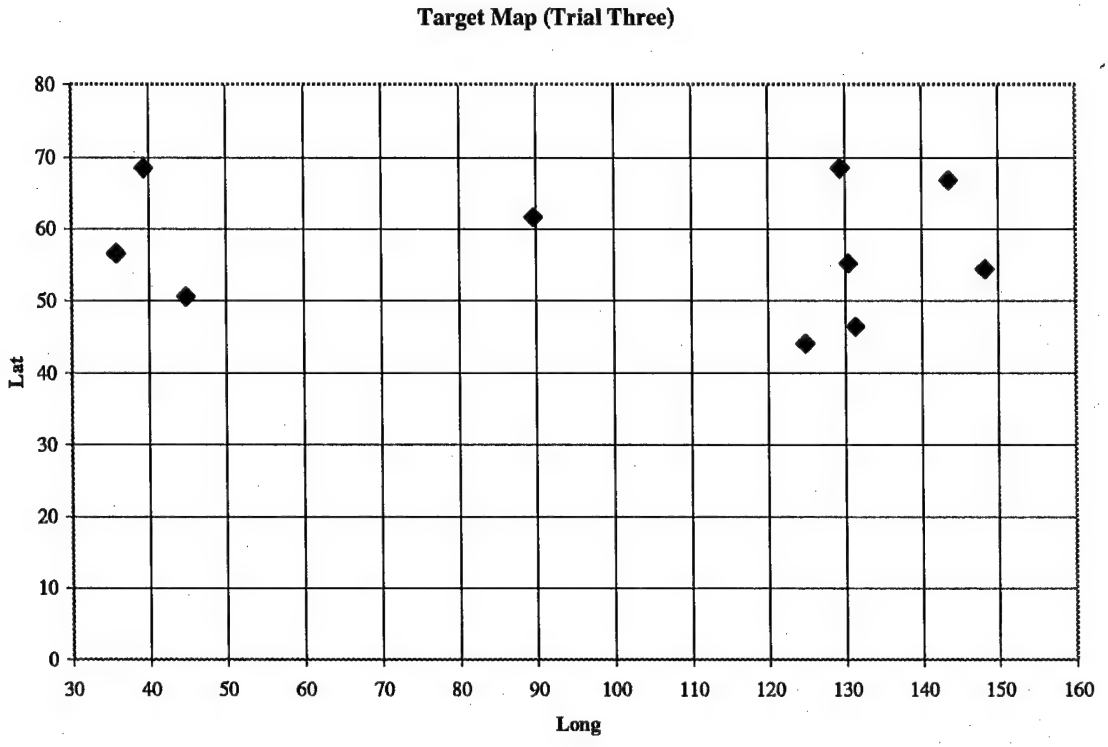


Figure 5-5. Target Map for Trial Three of the Ten Target Problem

It has been shown in Chapter 3 that this generator produces many good routes that move from north to south with backtracking. However, it also produces a large number of undesired routes. In fact, on the ten target problem this generator showed the poorest performance in both processing time and route reduction. Only three trials were performed due to the length of the processing time of the third trial. It took 22.5 hours to return nearly 350,000 routes. For this reason, the true backtracking generator was created

and testing of the north south walk with backtracking generator was discontinued. It is interesting to note the relationship between the north south walk with backtracking generator and the distance generator. For example, in the first test trial problem the distance generator produced 24,000 routes and the north south walk with backtracking generator only produced 15,000. On the third trial the distance generator performed much better producing only 1,662 routes compared to the 350,000 routes created by the north south walk with backtracking. The locations of the targets for trial three are seen in Figure 5-5. The figure shows that there are two distinct clusters of targets. So, in trial three there are fewer possible routes produced by the distance generator because of the distance separating the two clusters of targets. The performance of the distance generator on trial three and its relation to the true backtracking generator are discussed in more detail later in this chapter. How these differences effect the north south walk with backtracking generator is not as clear. The bottom line is the north south walk with backtracking produces undesired routes more frequently than the distance generator.

True Backtracking Generator

Like the distance generator, the true backtracking generator did reduce the number of routes produced. On the ten target problem, the average number of routes generated was 6,624. The associated reduction is about 0.4% of the possible routes and therefore it appears that the true backtracking generator performed better than the distance generator in reducing the number of routes created. A surprising result was the time required for the true backtracking generator to produce its routes. In one case, trial three of the ten target problem, the true backtracking generator created many more routes than the distance generator, but did so in less time. Because there are many more

operations required to create a route in the true backtracking generator than the distance generator, the true backtracking generator was expected to take longer than the distance generator. The reason is the true backtracking generator removes more targets from consideration earlier in the process of building routes, and therefore considers fewer routes than the distance generator. The distance generator considers more targets because the distance between any two targets has an upper limit. The greatest possible distance between the sample targets is the great circle between 40 degrees North, 30 degrees East and 70 degrees North and 160 degrees East. This distance is 3850 nautical miles and a distance this large was not seen often. If a distance this large had appeared more frequently the random numbers used to generate the target locations would have to be questioned. Since all distances are less than 3850, typically the maximum distance was not exceeded until several targets were assigned to a route. Thus, not as many possible routes were eliminated since targets were not removed from consideration early in the route formation process and a corresponding reduction in processing time was not observed when compared to the true backtracking generator.

Trial three is also important because it is one of the three trials where the true backtracking generator performance was worse than the distance generator's. Figure 5-5 is a map of the target locations for trial three. There are two distinct clusters of targets and one cluster has three targets and the other has six. It seems that due to the separation between the two groups, not as many routes can be eliminated once at a given target. This is especially true when the generator is at a target in the cluster of three. This leaves a larger number of routes that are considered feasible by the true backtracking generator. The distance generator performed better than expected on trial three. Remember each

route has to have eight targets. The size of the two clusters of targets was six and three, with a single isolated target between the two groups. This means that there were fewer ways to make routes of eight. If the route began in the three target cluster and then selected a target in the six target cluster it would have to return to the first cluster or the isolated target to make a route of eight. After it picked another target from the first cluster, it would have to return to the cluster of six to finish the route.

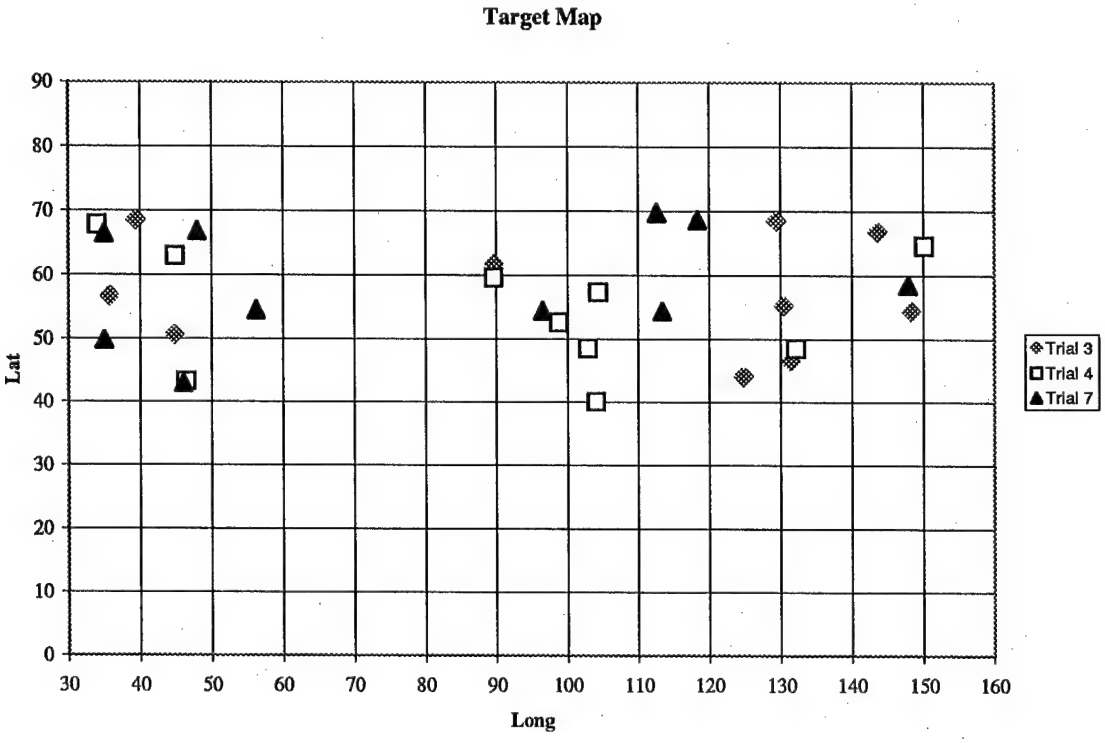


Figure 5-6. Target Map for Trials Three, Four and Seven

The distance separating the two groups is too large to allow many of these routes to be feasible. Actually, this is exactly what was expected from the distance generator. It prevents this crisscrossing, and the locations of these targets require crisscrossing to make routes of eight. This is also the case for trials four and seven. In these two trials, the distance generator outperformed the true backtracking generator, but to a lesser degree. Figure 5-6 is a map of the target locations for trials three, four, and seven. The

map shows that the three trials have two groups of targets, but the grouping of targets in trials four and seven are not as distinct as in trial three. This may account for the improved performance of the true backtracking generator in trials four and seven over trial three.

True Backtracking and Distance Generator Combined

Table 5-1. Ten Target Problem Summary of Results

Generator Statistics						
	Distance Generator		True Backtracking Generator		Combination	
Trial	Number of Routes	Processing Time (sec)	Number of Routes	Processing Time (sec)	Number of Routes	Processing Time (sec)
1	24120	17803	1718	937	725	939
2	46990	55485	19491	3811	2940	2244
3	1662	8793	18409	3698	223	1342
4	1130	6639	3543	1461	24	641
5	5308	7800	2063	878	232	731
6	8340	11735	1564	975	229	667
7	2326	6703	4975	1647	133	887
8	17952	17913	5160	1767	691	1400
9	9802	11111	5409	1799	309	1352
10	35224	35234	3912	1649	688	1582
Total	152854	179216	66244	18622	6194	11783
Average	15285	17922	6624	1862	619	1178
Variance	2.E+08	2.E+08	4.E+07	1.E+06	7.E+05	3.E+05

The combination of the true backtracking generator and the distance generator did a good job of reducing the number of routes considered on the ten target problem. The combination also ran fairly quickly. The average number of routes generated for the ten trials was 619 and the average processing time was 1178 seconds. In all but one trial case (trial two), less than 1,000 routes were generated. The reason the combination performed worse on trial two is because of the target clustering already discussed. This clustering of

targets will certainly affect the distance generator. Examining the number of routes produced by the distance generator for trial two confirms this fact. The total number of routes created is nearly three times the mean. Table F-1 lists the target locations, minimum and maximum values for latitude and longitude, and the latitude and longitude ranges for all ten trials. The longitude ranges for trial two and ten are significantly smaller than the longitude range of the other eight trials. Reduced range in longitude also identifies the degree of target clustering. Thus, the poorer performance of the distance and the combination generators on these two trials is easily understood.

Table 5-1 is a summary of the processing times and number of routes generated for the three best generators. The reduction in routes and the processing time of the combination of the distance and true backtracking generators seemed to indicate that it performs the best. This was expected, and as a result, it was selected for use with the larger sample problems. At first glance, the variance of the ten trials in Table 5-1 for each generator is very large. Large variances were expected because of the target generation procedure. Target locations were allowed to vary over a wide range and each problem only had ten targets. It seems logical then that the results would vary from trail to trail and therefore, the variance would be large. It was decided to conduct tests between each pair of generators to verify that there is enough evidence to suggest that the averages from each generator are statistically different. Due to the small sample size, the usual assumption of normally distributed data could not be made. For this reason, a nonparametric test is used. Computer software was used to calculate the statistics used. The software uses the Mann-Whitney *U* Test. *U* is obtain by ordering all observations from both samples according to their magnitude and counting the number of observations

in the first sample that precedes each observation in the second sample. Very large or very small values of U imply that the medians of the two samples are different. The test to compare the medians of the distance and true backtracking generators is used to illustrate the procedure. The null hypothesis being tested is that the medians are the same. The alternate hypothesis is the median of the distance generator is larger than the median of the true backtracking generator. The U statistic for the distance generator in this test was found to be 35. Testing at 95% confidence, the critical value for U is 27 [16:749]. In order to reject the null hypothesis, the U statistic must be less than the critical value for U. Since this is not true in this case, the null hypothesis cannot be rejected. More trials are needed to conclude the true backtracking generator outperforms the distance generator. Similar tests were performed to compare the distance generator to the combination and the true backtracking generator to the combination generator. These tests did show the expected result that the combination generator is the best method. It should be the case that the combination is different than the other two generators because it is doing both things at once. The results are summarized in Table 5-2.

Table 5-2. Results of Mann-Whitney *U* for Different Means

Generators	U Value	U Critical	P-value
Distance and True Backtracking	35	27	.137
Distance and Combination	3	27	.000
True Backtracking and Combination	3	27	.000

Large Sample Problems

The large sample problems were not tested on the same computer as the eight and ten target problems. The computers used for the large sample problems were older and their processing speeds were much slower than what is currently available. The reason the generators were not tried on faster computers is because even on fast computers, the results of the ten target problem indicated the generators would take a long time to run (several days) and computer resources were limited. Both the 200 target and 100 target problems were aborted before they could be finished. The 200 target problem ran for about a week and generated just under a million routes. This seems like good progress, but the first five places on the route were 1-2-4-5-10. This means that there were almost a million routes generated with the same two targets occupying the first two places on every route. This was not the kind of progress that was hoped for. The results were similar for the 100 target problem. A 50 target problem was also tried, but it was also too large.

Early in the testing of the distance generator, a problem with 50 targets and five targets per route was attempted on newer computer technology. There was no sorting of the targets like that tried with the other large sample problems. This problem did complete and generated about 2 million feasible routes out of about 254 million possible routes. It took three and a half days to complete. For this reason it is felt that a 25 target problem with eight targets per route could be completed on a faster computer. This problem has $4.36 \text{ X } 10^{10}$ possible routes which makes it about 200 times larger than the 50 target with five targets per route problem. It may seem more logical to try a problem

with 50 targets and eight targets per route, but it turns out this problem is about 85,000 times larger than the 50 target with five targets per route problem.

6. Conclusions and Recommendations

Introduction

Chapter 6 contains two major sections. The first section makes recommendations for future use of the results of this research. The second section discusses areas for future research to improve the route generation process.

Recommendations

Based on the results of Chapter 5, there is still a need for more information on the performance of the combination of the true backtracking and distance generators on faster computers. Before the combination is used on a real target database, two things need to be determined. What is the largest number of targets that can be used? This depends on the computer used and the location of the targets. Test problems of increasing size should be tested on the computer that will be used to generate routes. The idea is to find the largest possible problem that runs in a relatively short amount of time. This amount of time probably needs to be less than a few hours because the generator cannot handle problems using the entire target database in a single run. Based on the work done in this research, the problem size is most likely going to be between 10 and 25 targets per run. Due to this limitation, the combination generator will have to be run several times and this leads to the next question. How to divide the target database? Knowing something about the composition of current bomber routes would help in making this decision. This information is not available for this research, so two general approaches are suggested. One approach is to divide the targets into separate geographic regions by using lines of longitude and latitude. There may be a problem with using geographic regions. If the

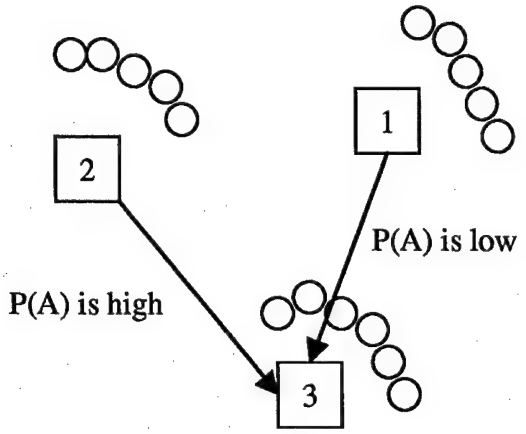
targets are not distributed evenly throughout the target area, then some regions may contain many targets and others regions may contain few or no targets. Some experimenting with the size of the geographic regions will be required to get the number of targets per region in between 10 and 25. An easier way to divide the target database would be to pick a given number of targets at random and place them in groups. As just mentioned, the size of these groups would depend on the speed of the computer used. The combination generator could be applied to these groups until all targets are covered by a certain number of routes. There are many possibilities here and choosing the best requires knowledge of the targets and the current routes. Once these two questions have been answered, the combination generator could be run to generate a pool of routes that are distance feasible.

Areas for Further Research

Methods for Reducing the Number of Routes Generated

There are other constraints on bomber routes that could be used to create new generators or improve the generators studied in this research. One example is the probability of survival of a bomber given that bomber is on a specified route. One way of calculating the probability of survival is to multiply the probabilities of arriving at each target together. The probability of arrival at a target depends on many things. Such as the direction of approach to that target. Since this is the case, the probability of arrival at a given target depends on what the previous target on the route is. For example, Figure 6-1 shows the probability of arrival at target 3 coming directly from target 2 which is not necessarily the same as the probability of arrival at target 3 coming directly from target 1.

Therefore, there is a square matrix similar to the distance matrix that describes the probability of arriving at each target from every other target.



○ Air Defense Radar Site
P(A) is the Probability of Arrival

Table 6-1. Diagram of How the Probability of Arrival at a Target Might Vary

This information could be used in the same way the distance matrix was used to generate routes. The only difference being that instead of summing the distance at every step in a route, the probabilities of arrival could be multiplied together to keep a running probability of survival for the route being created. If the addition of a target causes the probability of survival to drop below a specified minimum, that target would be removed from consideration. The problem with adding this to the combination of the true backtracking generator and the distance generator is it adds more mathematical operations to the process. Again, there is a tradeoff. If the reduction in routes is large enough, then adding the probability of arrival would be worthwhile. However, there is another way to use the probability of arrival information with the distance generator to create a pool of routes. There is a strong possibility that the probability of arrival at a

target directly from another target is zero or near zero. There may be several of these zero probabilities. This fact could be used to remove a distance from the distance matrix or change the distance between these two targets to zero, effectively making the distance matrix more sparse. For example, if the probability of arrival at target two from target one is zero, the distance between these two targets in the distance matrix could be changed to zero. The combination generator could be adapted to check for distances of zero and remove targets from consideration that are zero nautical miles from the current target. Many instances of zero or near zero probabilities of arrival could significantly reduce processing time.

In the end, the more factors that can be used to reduce the density of the distance matrix, the better the performance of all the generators will be. A quick test was done with the distance generator. A zero and one matrix the same size of the distance matrix was created from a discrete probability function. The probability of a zero occurring anywhere in the matrix was 0.1 and the probability of a one was 0.9. Once this matrix was created, zero entries in the zero and one matrix were copied into the same location in the corresponding distance matrix for trial one of the ten target problem. The distance generator was run on this new matrix and the performance was compared to the full distance matrix. Approximately half of the number of routes were produced in half of the time. This is encouraging considering that the density of the matrix was only reduced a small amount.

Modifying a Search Heuristic.

Chapter 3 included a short discussion of search heuristics that are used to find solutions to large and difficult integer programs like the VRP. Modifying any of these

heuristics would be one possible way of generating a pool of routes. If the bomber routing problem is formulated as a VRP as was done in Chapter 3, then these heuristics could be easily modified to produce feasible routes as opposed to solving the problem. There are several ways to modify these heuristics, but one modification seems inviting. As mentioned in Chapter 3, when using tabu search once a solution is generated, it is placed on a tabu list. Solutions that are placed on this list frequently could be tracked. If the tabu search heuristic is designed appropriately, being on the tabu list many times can mean that the solution has desirable characteristics in terms of the objective function. Of course, this depends on the objective function. In this case it would mean the solution produces a set of routes that minimizes the distance. Solutions that are on the tabu list most often could be saved and the routes from that solution placed in a pool of routes. Without going into a lengthy discussion on Genetic Algorithms (GA), it is possible to do something similar with GA. GA uses fitness functions to determine which solutions are allowed to be in or form the next generation of solutions. Fit solutions are similar to solutions placed on the tabu list frequently. For this reason, keeping track of fit solutions may be useful in generating a pool of routes.

Summary

The size of the bomber routing problem cannot be over emphasized. Any method used to find feasible routes is going to have advantages and disadvantages. The number of possible routes to examine for feasibility is huge. So huge that the number of resulting feasible routes is extremely large. In fact, the number of feasible routes is so large that even if they could all be determined, there would be too many of them to use in WAM. Keeping this in mind, the methods for generating bomber routes in this research can be

used to make a reasonably sized pool of routes needed by WAM to optimize strategic weapons assignment.

Appendix A.

Appendix A contains the Microsoft Excel Visual Basic macro used to create routes based solely on great circle distance.

```
Sub DistanceGenerator()

' This macro uses the great circle distance matrix to create
' feasible bomber routes between the targets for the delivery of
' gravity weapons.

  Dim x1 As Integer 'looping variable for the targets assigned
  Dim x2 As Integer 'to x1, x2, ..., x8 position on a route
  Dim x3 As Integer
  Dim x4 As Integer
  Dim x5 As Integer
  Dim x6 As Integer
  Dim x7 As Integer
  Dim x8 As Integer

  Dim Total As Long 'Total number of routes created
  Total = 0

  Dim Dis1 As Integer 'Distance between target 1 and 2,...
  Dim Dis2 As Integer
  Dim Dis3 As Integer
  Dim Dis4 As Integer
  Dim Dis5 As Integer
  Dim Dis6 As Integer
  Dim Dis7 As Integer

  Dim Starttime As Single 'Processing time variables
  Dim Endtime As Single
  Dim Totaltime As Single

  Dim Distance As Integer 'Maximum route distance
  Dim TotalD1 As Integer 'Total route length calculated after
  Dim TotalD2 As Integer 'route is created
  Dim TotalD3 As Integer
  Dim TotalD4 As Integer
  Dim TotalD5 As Integer
  Dim TotalD6 As Integer
  Dim TotalD7 As Integer
  Distance = 6200 'setting the maximum distance
```

```
Dim OutputSheet As String 'variable output information
Dim InputSheet As String 'varaible for input information
Dim Trial As String 'variable to keep track of test trail #

InputSheet = InputBox("Select input sheet", "Route Generator", _
"Sheet1") 'sets location of distance matrix

OutputSheet = InputBox("Select output sheet", "Route Generator", _
"Sheet1") 'sets location of output information

Trial = InputBox("Input the Trial Run Number", "RouteGenerator", _
"Trial 2") 'sets the trail number for output

Application.WindowState = xlMinimized
Starttime = Timer 'sets start time

  For x1 = 1 To 10
   For x2 = 1 To 10
    Sheets(InputSheet).Select
    Dis1 = Cells(x1 + 1, x2 + 1).Value
    TotalD1 = Dis1
    If Or x2 = x1 Then
     GoTo line13
    Else: GoTo line1
    End If
line1:  For x3 = 1 To 10
     Sheets(InputSheet).Select
     Dis2 = Cells(x2 + 1, x3 + 1).Value
     TotalD2 = TotalD1 + Dis2
     If x3 = x2 Or x3 = x1 Then
      GoTo line12
     Else: GoTo line2
     End If
line2:   For x4 = 1 To 10
      Sheets(InputSheet).Select
      Dis3 = Cells(x3 + 1, x4 + 1).Value
      TotalD3 = TotalD2 + Dis3
      If  TotalD3 > Distance Or x4 = x3 Or x4 = x2 Or x4 = x1 Then
       GoTo line11
      Else: GoTo line3
      End If
line3:    For x5 = 1 To 10
       Sheets(InputSheet).Select
       Dis4 = Cells(x4 + 1, x5 + 1).Value
       TotalD4 = TotalD3 + Dis4
```

```
            If TotalD4 > Distance Or x5 = x4 Or x5 = x3 Or x5 = x2 Or _
            x5 = x1 Then
              GoTo line10
            Else: GoTo line4
            End If
line4:       For x6 = 1 To 10
             Sheets(InputSheet).Select
             Dis5 = Cells(x5 + 1, x6 + 1).Value
             TotalD5 = TotalD4 + Dis5
             If TotalD5 > Distance Or x6 = x5 Or x6 = x4 Or x6 = x3 Or _
             x6 = x2 Or x6 = x1 Then
               GoTo line9
             Else: GoTo line5
             End If
line5:        For x7 = 1 To 10
              Sheets(InputSheet).Select
              Dis6 = Cells(x6 + 1, x7 + 1).Value
              TotalD6 = TotalD5 + Dis6
              If TotalD6 > Distance Or x7 = x6 Or _
              x7 = x5 Or x7 = x4 Or x7 = x3 Or x7 = x2 _
              Or x7 = x1 Then
                GoTo line8
              Else: GoTo line6
              End If
line6:           For x8 = 1 To 10
               Sheets(InputSheet).Select
               Dis7 = Cells(x7 + 1, x8 + 1).Value
               TotalD7 = TotalD6 + Dis7
               If TotalD7 > Distance Or x8 = x7 Or _
               x8 = x6 Or x8 = x5 Or x8 = x4 Or x8 = x3 Or _
               x8 = x2 Or x8 = x1 Then
                 GoTo line7
               Else:  Total = Total + 1
                 'Sheets("sheet10").Select
                 'Cells(Total + 4, 1).Value = Total
                 'Cells(Total + 4, 2).Value = x1
                 'Cells(Total + 4, 3).Value = x2
                 'Cells(Total + 4, 4).Value = x3
                 'Cells(Total + 4, 5).Value = x4
                 'Cells(Total + 4, 6).Value = x5
                 'Cells(Total + 4, 7).Value = x6
                 'Cells(Total + 4, 8).Value = x7
                 'Cells(Total + 4, 9).Value = x8
               End If
line7:          Next x8
line8:        Next x7
```

```
line9:         Next x6
line10:      Next x5
line11:    Next x4
line12:  Next x3
line13: Next x2
    Next x1

Endtime = Timer 'Calculate processing time
Totaltime = Endtime - Starttime

Sheets(OutputSheet).Select 'Output Resutlts
Cells(1, 1).Value = "Distance Filter Statistics" & Trial
Cells(2, 1).Value = "Total"
Cells(3, 1).Value = Total
Cells(2, 2).Value = "Processing Time "
Cells(3, 2).Value = Totaltime

Application.WindowState = xlNormal
MsgBox "The total number or routes generated is " & Total, vbOKOnly, "Route
Genrator"

End Sub
```

Appendix B.

Appendix B contains the Microsoft Excel Visual Basic macro which generates routes based on the true backtracking generator developed in Chapter 3.

```
Sub BackTracker()

' This macro creates bomber routes using an approach that allows the
' bomber to move in any direction except the 90 degrees directly behind the
' current flight path.

Dim x1 As Integer 'looping variables for targets and the position
Dim x2 As Integer 'on the route
Dim x3 As Integer
Dim x4 As Integer
Dim x5 As Integer
Dim x6 As Integer
Dim x7 As Integer
Dim x8 As Integer

Dim Total As Long  'total number of routes generated
Total = 0 'initialized

Dim Lat1 As Double 'target latitude variables
Dim Lat2 As Double
Dim Lat3 As Double
Dim Lat4 As Double
Dim Lat5 As Double
Dim Lat6 As Double
Dim Lat7 As Double
Dim Lat8 As Double

Dim Long1 As Double 'target longitude variables
Dim Long2 As Double
Dim Long3 As Double
Dim Long4 As Double
Dim Long5 As Double
Dim Long6 As Double
Dim Long7 As Double
Dim Long8 As Double

Dim Slope1 As Double 'variable for slope of current flight path
Dim Slope2 As Double
Dim Slope3 As Double
```

```
Dim Slope4 As Double
Dim Slope5 As Double
Dim Slope6 As Double
Dim Slope7 As Double

Dim TestSlopeLine1 As Double 'variable to hold slope +45 degree value
Dim TestSlopeLine2 As Double 'variable to hold slope -45 degree value

Dim Starttime As Single 'variables for calculating processing time
Dim Endtime As Single
Dim Totaltime As Single

Dim OutputSheet As String 'variables for input/output
Dim InputSheet As String
Dim Trial As String

InputSheet = InputBox("Select input sheet", "Route Generator", _
"Sheet1") 'set input target location information

OutputSheet = InputBox("Select output sheet", "Route Generator", _
"Sheet1") 'set output information location

Trial = InputBox("Input the Trial Run Number", "RouteGenerator", _
"Trial 1") 'set test trial number

Application.WindowState = xlMinimized
Starttime = Timer

    For x1 = 1 To 10 'generate routes
      Sheets(InputSheet).Select
      Lat1 = Cells(x1 + 1, 2).Value
      Long1 = Cells(x1 + 1, 3).Value
      For x2 = 1 To 10
      Sheets(InputSheet).Select
      Lat2 = Cells(x2 + 1, 2).Value
      Long2 = Cells(x2 + 1, 3).Value
      If x2 = x1 Then
        GoTo line13
      End If
line1:  For x3 = 1 To 10
        Sheets(InputSheet).Select
        Lat3 = Cells(x3 + 1, 2).Value
        Long3 = Cells(x3 + 1, 3).Value
        If x3 = x2 Or x3 = x1 Then
          GoTo line12
        End If
```

```
Slope1 = (Lat2 - Lat1) / (Long2 - Long1)
TestSlopeLine1 = (Slope1 + 1) / (1 - Slope1)
TestSlopeLine2 = (Slope1 - 1) / (1 + Slope1)
If -1# <= Slope1 And Slope1 <= 1# Then
 If Long2 <= Long1 Then
  If Lat3 >= Lat2 Then
   If Lat3 <= TestSlopeLine1 * (Long3 - Long2) + Lat2 Then
    GoTo line12
   Else: GoTo line2
   End If
  ElseIf Lat3 < Lat2 Then
   If Lat3 >= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
    GoTo line12
   Else: GoTo line2
   End If
  End If
 ElseIf Long2 > Long1 Then
  If Lat3 >= Lat2 Then
   If Lat3 <= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
    GoTo line12
   Else: GoTo line2
   End If
  ElseIf Lat3 < Lat2 Then
   If Lat3 >= TestSlopeLine1 * (Long3 - Long2) + Lat2 Then
    GoTo line12
   Else: GoTo line2
   End If
  End If
 End If
ElseIf Slope1 > 1# Then
 If Lat2 <= Lat1 Then
  If Lat3 >= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
   GoTo line12
  Else: GoTo line2
  End If
 ElseIf Lat2 > Lat1 Then
  If Lat3 <= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
   GoTo line12
  Else: GoTo line2
  End If
 End If
ElseIf Slope1 < -1# Then
 If Lat2 <= Lat1 Then
  If Lat3 >= TestSlopeLine1 * (Long3 - Long2) + Lat2 Then
   GoTo line12
  Else: GoTo line2
```

```
        End If
       ElseIf Lat2 > Lat1 Then
        If Lat3 <= TestSlopeLine1 * (Long3 + Long2) + Lat2 Then
         GoTo line12
        Else: GoTo line2
        End If
       End If
      End If
line2:  For x4 = 1 To 10
       Sheets(InputSheet).Select
       Lat4 = Cells(x4 + 1, 2).Value
       Long4 = Cells(x4 + 1, 3).Value
       If x4 = x3 Or x4 = x2 Or x4 = x1 Then
        GoTo line11
       End If
       Slope2 = (Lat3 - Lat2) / (Long3 - Long2)
       TestSlopeLine1 = (Slope2 + 1) / (1 - Slope2)
       TestSlopeLine2 = (Slope2 - 1) / (1 + Slope2)
       If -1# <= Slope2 And Slope2 <= 1# Then
        If Long3 <= Long2 Then
         If Lat4 >= Lat3 Then
          If Lat4 <= TestSlopeLine1 * (Long4 - Long3) + Lat3 Then
           GoTo line11
          Else: GoTo line3
          End If
         ElseIf Lat4 < Lat3 Then
          If Lat4 >= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
           GoTo line11
          Else: GoTo line3
          End If
         End If
        ElseIf Long3 > Long2 Then
         If Lat4 >= Lat3 Then
          If Lat4 <= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
           GoTo line11
          Else: GoTo line3
          End If
         ElseIf Lat4 < Lat3 Then
          If Lat4 >= TestSlopeLine1 * (Long4 - Long3) + Lat3 Then
           GoTo line11
          Else: GoTo line3
          End If
         End If
        End If
        ElseIf Slope2 > 1# Then
         If Lat3 <= Lat2 Then
```

```
            If Lat4 >= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
              GoTo line11
            Else: GoTo line3
            End If
           ElseIf Lat3 > Lat2 Then
            If Lat4 <= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
              GoTo line11
            Else: GoTo line3
            End If
           End If
           ElseIf Slope2 < -1# Then
            If Lat3 <= Lat2 Then
              If Lat4 >= TestSlopeLine1 * (Long4 - Long3) + Lat3 Then
                GoTo line11
              Else: GoTo line3
              End If
            ElseIf Lat3 > Lat2 Then
              If Lat4 <= TestSlopeLine1 * (Long4 + Long3) + Lat3 Then
                GoTo line11
              Else: GoTo line3
              End If
            End If
           End If
line3:      For x5 = 1 To 10
           Sheets(InputSheet).Select
           Lat5 = Cells(x5 + 1, 2).Value
           Long5 = Cells(x5 + 1, 3).Value
           If x5 = x4 Or x5 = x3 Or x5 = x2 Or x5 = x1 Then
            GoTo line10
           End If
           Slope3 = (Lat4 - Lat3) / (Long4 - Long3)
           TestSlopeLine1 = (Slope3 + 1) / (1 - Slope3)
           TestSlopeLine2 = (Slope3 - 1) / (1 + Slope3)
           If -1# <= Slope3 And Slope3 <= 1# Then
            If Long4 <= Long3 Then
              If Lat5 >= Lat4 Then
                If Lat5 <= TestSlopeLine1 * (Long5 - Long4) + Lat4 Then
                  GoTo line10
                Else: GoTo line4
                End If
              ElseIf Lat5 < Lat4 Then
                If Lat5 >= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
                  GoTo line10
                Else: GoTo line4
                End If
              End If
```

```
        ElseIf Long4 > Long3 Then
          If Lat5 >= Lat4 Then
            If Lat5 <= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
              GoTo line10
            Else: GoTo line4
            End If
          ElseIf Lat5 < Lat4 Then
            If Lat5 >= TestSlopeLine1 * (Long5 - Long4) + Lat4 Then
              GoTo line10
            Else: GoTo line4
            End If
          End If
        End If
      ElseIf Slope3 > 1# Then
        If Lat4 <= Lat3 Then
          If Lat5 >= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
            GoTo line10
          Else: GoTo line4
          End If
        ElseIf Lat4 > Lat3 Then
          If Lat5 <= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
            GoTo line10
          Else: GoTo line4
          End If
        End If
      ElseIf Slope3 < -1# Then
        If Lat4 <= Lat3 Then
          If Lat5 >= TestSlopeLine1 * (Long5 - Long4) + Lat4 Then
            GoTo line10
          Else: GoTo line4
          End If
        ElseIf Lat4 > Lat3 Then
          If Lat5 <= TestSlopeLine1 * (Long5 + Long4) + Lat4 Then
            GoTo line10
          Else: GoTo line4
          End If
        End If
      End If
line4:    For x6 = 1 To 10
      Sheets(InputSheet).Select
      Lat6 = Cells(x6 + 1, 2).Value
      Long6 = Cells(x6 + 1, 3).Value
      If x6 = x5 Or x6 = x4 Or x6 = x3 Or x6 = x2 _
      Or x6 = x1 Then
        GoTo line9
      End If
```

```
Slope4 = (Lat5 - Lat4) / (Long5 - Long4)
TestSlopeLine1 = (Slope4 + 1) / (1 - Slope4)
TestSlopeLine2 = (Slope4 - 1) / (1 + Slope4)
If -1# <= Slope4 And Slope4 <= 1# Then
 If Long5 <= Long4 Then
  If Lat6 >= Lat5 Then
   If Lat6 <= TestSlopeLine1 * (Long6 - Long5) + Lat5 Then
    GoTo line9
   Else: GoTo line5
   End If
  ElseIf Lat6 < Lat5 Then
   If Lat6 >= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
    GoTo line9
   Else: GoTo line5
   End If
  End If
 ElseIf Long5 > Long4 Then
  If Lat6 >= Lat5 Then
   If Lat6 <= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
    GoTo line9
   Else: GoTo line5
   End If
  ElseIf Lat6 < Lat5 Then
   If Lat6 >= TestSlopeLine1 * (Long6 - Long5) + Lat5 Then
    GoTo line9
   Else: GoTo line5
   End If
  End If
 End If
ElseIf Slope4 > 1# Then
 If Lat5 <= Lat4 Then
  If Lat6 >= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
   GoTo line9
  Else: GoTo line5
  End If
 ElseIf Lat5 > Lat4 Then
  If Lat6 <= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
   GoTo line9
  Else: GoTo line5
  End If
 End If
ElseIf Slope4 < -1# Then
 If Lat5 <= Lat4 Then
  If Lat6 >= TestSlopeLine1 * (Long6 - Long5) + Lat5 Then
   GoTo line9
  Else: GoTo line5
```

```
            End If
           ElseIf Lat5 > Lat4 Then
            If Lat6 <= TestSlopeLine1 * (Long6 + Long5) + Lat5 Then
             GoTo line9
            Else: GoTo line5
            End If
           End If
          End If
line5:      For x7 = 1 To 10
          Sheets(InputSheet).Select
          Lat7 = Cells(x7 + 1, 2).Value
          Long7 = Cells(x7 + 1, 3).Value
          If x7 = x6 Or x7 = x5 Or x7 = x4 Or _
          x7 = x3 Or x7 = x2 Or x7 = x1 Then
           GoTo line8
          End If
          Slope5 = (Lat6 - Lat5) / (Long6 - Long5)
          TestSlopeLine1 = (Slope5 + 1) / (1 - Slope5)
          TestSlopeLine2 = (Slope5 - 1) / (1 + Slope5)
          If -1# <= Slope5 And Slope5 <= 1# Then
           If Long6 <= Long5 Then
            If Lat7 >= Lat6 Then
             If Lat7 <= TestSlopeLine1 * (Long7 - Long6) + Lat6 Then
              GoTo line8
             Else: GoTo line6
             End If
            ElseIf Lat7 < Lat6 Then
             If Lat7 >= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
              GoTo line8
             Else: GoTo line6
             End If
            End If
           ElseIf Long6 > Long5 Then
            If Lat7 >= Lat6 Then
             If Lat7 <= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
              GoTo line8
             Else: GoTo line6
             End If
            ElseIf Lat7 < Lat6 Then
             If Lat7 >= TestSlopeLine1 * (Long7 - Long6) + Lat6 Then
              GoTo line8
             Else: GoTo line6
             End If
            End If
           End If
          ElseIf Slope5 > 1# Then
```

```
                If Lat6 <= Lat5 Then
                 If Lat7 >= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
                  GoTo line8
                 Else: GoTo line6
                 End If
                ElseIf Lat6 > Lat5 Then
                 If Lat7 <= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
                  GoTo line8
                 Else: GoTo line6
                 End If
                End If
               ElseIf Slope5 < -1# Then
                If Lat6 <= Lat5 Then
                 If Lat7 >= TestSlopeLine1 * (Long7 - Long6) + Lat6 Then
                  GoTo line8
                 Else: GoTo line6
                 End If
                ElseIf Lat6 > Lat5 Then
                 If Lat7 <= TestSlopeLine1 * (Long7 + Long6) + Lat6 Then
                  GoTo line8
                 Else: GoTo line6
                 End If
                End If
               End If
line6:           For x8 = 1 To 10
                Sheets(InputSheet).Select
                Lat8 = Cells(x8 + 1, 2).Value
                Long8 = Cells(x8 + 1, 3).Value
                If x8 = x7 Or x8 = x6 Or x8 = x5 Or _
                x8 = x4 Or x8 = x3 Or x8 = x2 Or _
                x8 = x1 Then
                 GoTo line7
                End If
                Slope6 = (Lat7 - Lat6) / (Long7 - Long6)
                TestSlopeLine1 = (Slope6 + 1) / (1 - Slope6)
                TestSlopeLine2 = (Slope6 - 1) / (1 + Slope6)
                If -1# <= Slope6 And Slope6 <= 1# Then
                 If Long7 <= Long6 Then
                  If Lat8 >= Lat7 Then
                   If Lat8 <= TestSlopeLine1 * (Long8 - Long7) + Lat7 Then
                    GoTo line7
                   End If
                  ElseIf Lat8 < Lat7 Then
                   If Lat8 >= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
                    GoTo line7
                   End If
```

```
            End If
          ElseIf Long7 > Long6 Then
            If Lat8 >= Lat7 Then
              If Lat8 <= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
                GoTo line7
              End If
            ElseIf Lat8 < Lat7 Then
              If Lat8 >= TestSlopeLine1 * (Long8 - Long7) + Lat7 Then
                GoTo line7
              End If
            End If
          End If
        ElseIf Slope6 > 1# Then
          If Lat7 <= Lat6 Then
            If Lat8 >= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
              GoTo line7
            End If
          ElseIf Lat7 > Lat6 Then
            If Lat8 <= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
              GoTo line7
            End If
          End If
        ElseIf Slope6 < -1# Then
          If Lat7 <= Lat6 Then
            If Lat8 >= TestSlopeLine1 * (Long8 - Long7) + Lat7 Then
              GoTo line7
            End If
          ElseIf Lat7 > Lat6 Then
            If Lat8 <= TestSlopeLine1 * (Long8 + Long7) + Lat7 Then
              GoTo line7
            End If
          End If
        End If
          Total = Total + 1
          'Sheets(OutputSheet).Select
          'Cells(Total + 4, 1).Value = Total
          'Cells(Total + 4, 2).Value = x1
          'Cells(Total + 4, 3).Value = x2
          'Cells(Total + 4, 4).Value = x3
          'Cells(Total + 4, 5).Value = x4
          'Cells(Total + 4, 6).Value = x5
          'Cells(Total + 4, 7).Value = x6
          'Cells(Total + 4, 8).Value = x7
          'Cells(Total + 4, 9).Value = x8
line7:      Next x8
line8:    Next x7
```

```
line9:              Next x6
line10:           Next x5
line11:         Next x4
line12:       Next x3
line13:     Next x2
        Next x1

Endtime = Timer 'calculate processing time
Totaltime = Endtime - Starttime

Sheets(OutputSheet).Select 'output results
Cells(1, 1).Value = "270 Degree Backtracking (only) Filter Statistics " & Trial
Cells(2, 1).Value = "Total"
Cells(3, 1).Value = Total
Cells(2, 2).Value = "Processing Time "
Cells(3, 2).Value = Totaltime

Application.WindowState = xlNormal
MsgBox "The total number or routes generated is " & Total, vbOKOnly, "Route
Genrator"

End Sub
```

Appendix C.

Appendix C contains the Microsoft Excel Visual Basic macro used to generate routes by applying a combination of the true backtracking and distance generators.

```
Sub BackTrackingGenerator()

    Dim x1 As Integer
    Dim x2 As Integer
    Dim x3 As Integer
    Dim x4 As Integer
    Dim x5 As Integer
    Dim x6 As Integer
    Dim x7 As Integer
    Dim x8 As Integer

    Dim Total As Long
    Total = 0

    Dim Lat1 As Double
    Dim Lat2 As Double
    Dim Lat3 As Double
    Dim Lat4 As Double
    Dim Lat5 As Double
    Dim Lat6 As Double
    Dim Lat7 As Double
    Dim Lat8 As Double

    Dim Long1 As Double
    Dim Long2 As Double
    Dim Long3 As Double
    Dim Long4 As Double
    Dim Long5 As Double
    Dim Long6 As Double
    Dim Long7 As Double
    Dim Long8 As Double

    Dim Dis1 As Integer
    Dim Dis2 As Integer
    Dim Dis3 As Integer
    Dim Dis4 As Integer
    Dim Dis5 As Integer
    Dim Dis6 As Integer
    Dim Dis7 As Integer
```

```
Dim Slope1 As Double
Dim Slope2 As Double
Dim Slope3 As Double
Dim Slope4 As Double
Dim Slope5 As Double
Dim Slope6 As Double
Dim Slope7 As Double

Dim TestSlopeLine1 As Double
Dim TestSlopeLine2 As Double

Dim Distance As Integer
Dim TotalD As Integer
Dim TotalD1 As Integer
Dim TotalD2 As Integer
Dim TotalD3 As Integer
Dim TotalD4 As Integer
Dim TotalD5 As Integer
Dim TotalD6 As Integer
Distance = 6200

Dim Starttime As Single
Dim Endtime As Single
Dim Totaltime As Single

Dim OutputSheet As String
Dim InputSheet As String
Dim Trial As String

InputSheet = InputBox("Select input sheet", "Route Generator", _
"Sheet1")

OutputSheet = InputBox("Select output sheet", "Route Generator", _
"Sheet1")

Trial = InputBox("Input the Trial Run Number", "RouteGenerator", _
"Trial 2")

Application.WindowState = xlMinimized
Starttime = Timer

  For x1 = 1 To 10
    Sheets(InputSheet).Select
    Lat1 = Cells(x1 + 13, 2).Value
    Long1 = Cells(x1 + 13, 3).Value
```

```
      For x2 = 1 To 10
       Sheets(InputSheet).Select
       Lat2 = Cells(x2 + 13, 2).Value
       Long2 = Cells(x2 + 13, 3).Value
       Dis1 = Cells(x1 + 1, x2 + 1).Value
       TotalD = Dis1
       If x2 = x1 Then
        GoTo line13
       End If
line1:   For x3 = 1 To 10
       Sheets(InputSheet).Select
       Lat3 = Cells(x3 + 13, 2).Value
       Long3 = Cells(x3 + 13, 3).Value
       Dis2 = Cells(x2 + 1, x3 + 1).Value
       TotalD1 = TotalD + Dis2
       If x3 = x2 Or x3 = x1 Or TotalD1 >= Distance Then
        GoTo line12
       End If
       Slope1 = (Lat2 - Lat1) / (Long2 - Long1)
       TestSlopeLine1 = (Slope1 + 1) / (1 - Slope1)
       TestSlopeLine2 = (Slope1 - 1) / (1 + Slope1)
       If -1# <= Slope1 And Slope1 <= 1# Then
        If Long2 <= Long1 Then
         If Lat3 >= Lat2 Then
          If Lat3 <= TestSlopeLine1 * (Long3 - Long2) + Lat2 Then
           GoTo line12
           Else: GoTo line2
          End If
         ElseIf Lat3 < Lat2 Then
          If Lat3 >= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
           GoTo line12
           Else: GoTo line2
          End If
         End If
        ElseIf Long2 > Long1 Then
         If Lat3 >= Lat2 Then
          If Lat3 <= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
           GoTo line12
           Else: GoTo line2
          End If
         ElseIf Lat3 < Lat2 Then
          If Lat3 >= TestSlopeLine1 * (Long3 - Long2) + Lat2 Then
           GoTo line12
           Else: GoTo line2
          End If
         End If
```

```
        End If
       ElseIf Slope1 > 1# Then
        If Lat2 <= Lat1 Then
         If Lat3 >= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
          GoTo line12
          Else: GoTo line2
         End If
        ElseIf Lat2 > Lat1 Then
         If Lat3 <= TestSlopeLine2 * (Long3 - Long2) + Lat2 Then
          GoTo line12
          Else: GoTo line2
         End If
        End If
       ElseIf Slope1 < -1# Then
        If Lat2 <= Lat1 Then
         If Lat3 >= TestSlopeLine1 * (Long3 - Long2) + Lat2 Then
          GoTo line12
          Else: GoTo line2
         End If
        ElseIf Lat2 > Lat1 Then
         If Lat3 <= TestSlopeLine2 * (Long3 + Long2) + Lat2 Then
          GoTo line12
          Else: GoTo line2
         End If
        End If
       End If
line2:   For x4 = 1 To 10
        Sheets(InputSheet).Select
        Lat4 = Cells(x4 + 13, 2).Value
        Long4 = Cells(x4 + 13, 3).Value
        Dis3 = Cells(x3 + 1, x4 + 1).Value
        TotalD2 = TotalD1 + Dis3
        If x4 = x3 Or x4 = x2 Or x4 = x1 Or TotalD2 _
        >= Distance Then
          GoTo line11
        End If
        Slope2 = (Lat3 - Lat2) / (Long3 - Long2)
        TestSlopeLine1 = (Slope2 + 1) / (1 - Slope2)
        TestSlopeLine2 = (Slope2 - 1) / (1 + Slope2)
        If -1# <= Slope2 And Slope2 <= 1# Then
          If Long3 <= Long2 Then
            If Lat4 >= Lat3 Then
              If Lat4 <= TestSlopeLine1 * (Long4 - Long3) + Lat3 Then
                GoTo line11
              Else: GoTo line3
              End If
```

```
        ElseIf Lat4 < Lat3 Then
          If Lat4 >= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
            GoTo line11
          Else: GoTo line3
          End If
        End If
      ElseIf Long3 > Long2 Then
        If Lat4 >= Lat3 Then
          If Lat4 <= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
            GoTo line11
          Else: GoTo line3
          End If
        ElseIf Lat4 < Lat3 Then
          If Lat4 >= TestSlopeLine1 * (Long4 - Long3) + Lat3 Then
            GoTo line11
          Else: GoTo line3
          End If
        End If
      End If
    ElseIf Slope2 > 1# Then
      If Lat3 <= Lat2 Then
        If Lat4 >= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
          GoTo line11
        Else: GoTo line3
        End If
      ElseIf Lat3 > Lat2 Then
        If Lat4 <= TestSlopeLine2 * (Long4 - Long3) + Lat3 Then
          GoTo line11
        Else: GoTo line3
        End If
      End If
    ElseIf Slope2 < -1# Then
      If Lat3 <= Lat2 Then
        If Lat4 >= TestSlopeLine1 * (Long4 - Long3) + Lat3 Then
          GoTo line11
        Else: GoTo line3
        End If
      ElseIf Lat3 > Lat2 Then
        If Lat4 <= TestSlopeLine1 * (Long4 + Long3) + Lat3 Then
          GoTo line11
        Else: GoTo line3
        End If
      End If
    End If
line3:  For x5 = 1 To 10
    Sheets(InputSheet).Select
```

```
Lat5 = Cells(x5 + 13, 2).Value
Long5 = Cells(x5 + 13, 3).Value
Dis4 = Cells(x4 + 1, x5 + 1).Value
TotalD3 = TotalD2 + Dis4
If x5 = x4 Or x5 = x3 Or x5 = x2 Or x5 = x1 _
Or TotalD3 >= Distance Then
 GoTo line10
End If
Slope3 = (Lat4 - Lat3) / (Long4 - Long3)
TestSlopeLine1 = (Slope3 + 1) / (1 - Slope3)
TestSlopeLine2 = (Slope3 - 1) / (1 + Slope3)
If -1# <= Slope3 And Slope3 <= 1# Then
 If Long4 <= Long3 Then
  If Lat5 >= Lat4 Then
   If Lat5 <= TestSlopeLine1 * (Long5 - Long4) + Lat4 Then
    GoTo line10
   Else: GoTo line4
   End If
  ElseIf Lat5 < Lat4 Then
   If Lat5 >= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
    GoTo line10
   Else: GoTo line4
   End If
  End If
 ElseIf Long4 > Long3 Then
  If Lat5 >= Lat4 Then
   If Lat5 <= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
    GoTo line10
   Else: GoTo line4
   End If
  ElseIf Lat5 < Lat4 Then
   If Lat5 >= TestSlopeLine1 * (Long5 - Long4) + Lat4 Then
    GoTo line10
   Else: GoTo line4
   End If
  End If
 End If
ElseIf Slope3 > 1# Then
 If Lat4 <= Lat3 Then
  If Lat5 >= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
   GoTo line10
  Else: GoTo line4
  End If
 ElseIf Lat4 > Lat3 Then
  If Lat5 <= TestSlopeLine2 * (Long5 - Long4) + Lat4 Then
   GoTo line10
```

```
            Else: GoTo line4
            End If
           End If
          ElseIf Slope3 < -1# Then
           If Lat4 <= Lat3 Then
            If Lat5 >= TestSlopeLine1 * (Long5 - Long4) + Lat4 Then
             GoTo line10
            Else: GoTo line4
            End If
           ElseIf Lat4 > Lat3 Then
            If Lat5 <= TestSlopeLine1 * (Long5 + Long4) + Lat4 Then
             GoTo line10
            Else: GoTo line4
            End If
           End If
          End If
line4:     For x6 = 1 To 10
          Sheets(InputSheet).Select
          Lat6 = Cells(x6 + 13, 2).Value
          Long6 = Cells(x6 + 13, 3).Value
          Dis5 = Cells(x5 + 1, x6 + 1).Value
          TotalD4 = TotalD3 + Dis5
          If x6 = x5 Or x6 = x4 Or x6 = x3 Or x6 = x2 _
          Or x6 = x1 Or TotalD4 >= Distance Then
           GoTo line9
          End If
          Slope4 = (Lat5 - Lat4) / (Long5 - Long4)
          TestSlopeLine1 = (Slope4 + 1) / (1 - Slope4)
          TestSlopeLine2 = (Slope4 - 1) / (1 + Slope4)
          If -1# <= Slope4 And Slope4 <= 1# Then
           If Long5 <= Long4 Then
            If Lat6 >= Lat5 Then
             If Lat6 <= TestSlopeLine1 * (Long6 - Long5) + Lat5 Then
              GoTo line9
             Else: GoTo line5
             End If
            ElseIf Lat6 < Lat5 Then
             If Lat6 >= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
              GoTo line9
             Else: GoTo line5
             End If
            End If
           ElseIf Long5 > Long4 Then
            If Lat6 >= Lat5 Then
             If Lat6 <= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
              GoTo line9
```

```
            Else: GoTo line5
            End If
           ElseIf Lat6 < Lat5 Then
            If Lat6 >= TestSlopeLine1 * (Long6 - Long5) + Lat5 Then
             GoTo line9
            Else: GoTo line5
            End If
           End If
          End If
         ElseIf Slope4 > 1# Then
          If Lat5 <= Lat4 Then
           If Lat6 >= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
            GoTo line9
           Else: GoTo line5
           End If
          ElseIf Lat5 > Lat4 Then
           If Lat6 <= TestSlopeLine2 * (Long6 - Long5) + Lat5 Then
            GoTo line9
           Else: GoTo line5
           End If
          End If
         ElseIf Slope4 < -1# Then
          If Lat5 <= Lat4 Then
           If Lat6 >= TestSlopeLine1 * (Long6 - Long5) + Lat5 Then
            GoTo line9
           Else: GoTo line5
           End If
          ElseIf Lat5 > Lat4 Then
           If Lat6 <= TestSlopeLine1 * (Long6 + Long5) + Lat5 Then
            GoTo line9
           Else: GoTo line5
           End If
          End If
         End If
line5:    For x7 = 1 To 10
          Sheets(InputSheet).Select
          Lat7 = Cells(x7 + 13, 2).Value
          Long7 = Cells(x7 + 13, 3).Value
          Dis6 = Cells(x6 + 1, x7 + 1).Value
          TotalD5 = TotalD4 + Dis6
          If x7 = x6 Or x7 = x5 Or x7 = x4 Or x7 = x3 _
          Or x7 = x2 Or x7 = x1 Or TotalD5 >= Distance _
          Then
           GoTo line8
          End If
          Slope5 = (Lat6 - Lat5) / (Long6 - Long5)
```

```
TestSlopeLine1 = (Slope5 + 1) / (1 - Slope5)
TestSlopeLine2 = (Slope5 - 1) / (1 + Slope5)
If -1# <= Slope5 And Slope5 <= 1# Then
 If Long6 <= Long5 Then
  If Lat7 >= Lat6 Then
   If Lat7 <= TestSlopeLine1 * (Long7 - Long6) + Lat6 Then
    GoTo line8
   Else: GoTo line6
   End If
  ElseIf Lat7 < Lat6 Then
   If Lat7 >= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
    GoTo line8
   Else: GoTo line6
   End If
  End If
 ElseIf Long6 > Long5 Then
  If Lat7 >= Lat6 Then
   If Lat7 <= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
    GoTo line8
   Else: GoTo line6
   End If
  ElseIf Lat7 < Lat6 Then
   If Lat7 >= TestSlopeLine1 * (Long7 - Long6) + Lat6 Then
    GoTo line8
   Else: GoTo line6
   End If
  End If
 End If
ElseIf Slope5 > 1# Then
 If Lat6 <= Lat5 Then
  If Lat7 >= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
   GoTo line8
  Else: GoTo line6
  End If
 ElseIf Lat6 > Lat5 Then
  If Lat7 <= TestSlopeLine2 * (Long7 - Long6) + Lat6 Then
   GoTo line8
  Else: GoTo line6
  End If
 End If
ElseIf Slope5 < -1# Then
 If Lat6 <= Lat5 Then
  If Lat7 >= TestSlopeLine1 * (Long7 - Long6) + Lat6 Then
   GoTo line8
  Else: GoTo line6
  End If
```

```
            ElseIf Lat6 > Lat5 Then
              If Lat7 <= TestSlopeLine1 * (Long7 + Long6) + Lat6 Then
                GoTo line8
              Else: GoTo line6
              End If
            End If
          End If
line6:      For x8 = 1 To 10
          Sheets(InputSheet).Select
          Lat8 = Cells(x8 + 13, 2).Value
          Long8 = Cells(x8 + 13, 3).Value
          Dis7 = Cells(x7 + 1, x8 + 1).Value
          TotalD6 = TotalD5 + Dis7
          If x8 = x7 Or x8 = x6 Or x8 = x5 Or x8 = x4 _
          Or x8 = x3 Or x8 = x2 Or x8 = x1 Or TotalD6 _
          >= Distance Then
            GoTo line7
          End If
          Slope6 = (Lat7 - Lat6) / (Long7 - Long6)
          TestSlopeLine1 = (Slope6 + 1) / (1 - Slope6)
          TestSlopeLine2 = (Slope6 - 1) / (1 + Slope6)
          If -1# <= Slope6 And Slope6 <= 1# Then
            If Long7 <= Long6 Then
              If Lat8 >= Lat7 Then
                If Lat8 <= TestSlopeLine1 * (Long8 - Long7) + Lat7 Then
                  GoTo line7
                End If
              ElseIf Lat8 < Lat7 Then
                If Lat8 >= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
                  GoTo line7
                End If
              End If
            ElseIf Long7 > Long6 Then
              If Lat8 >= Lat7 Then
                If Lat8 <= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
                  GoTo line7
                End If
              ElseIf Lat8 < Lat7 Then
                If Lat8 >= TestSlopeLine1 * (Long8 - Long7) + Lat7 Then
                  GoTo line7
                End If
              End If
            End If
          ElseIf Slope6 > 1# Then
            If Lat7 <= Lat6 Then
              If Lat8 >= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
```

```
                    GoTo line7
                  End If
                 ElseIf Lat7 > Lat6 Then
                  If Lat8 <= TestSlopeLine2 * (Long8 - Long7) + Lat7 Then
                   GoTo line7
                  End If
                 End If
                ElseIf Slope6 < -1# Then
                 If Lat7 <= Lat6 Then
                  If Lat8 >= TestSlopeLine1 * (Long8 - Long7) + Lat7 Then
                   GoTo line7
                  End If
                 ElseIf Lat7 > Lat6 Then
                  If Lat8 <= TestSlopeLine1 * (Long8 + Long7) + Lat7 Then
                   GoTo line7
                  End If
                 End If
                End If
                 Total = Total + 1
                 'Sheets(OutputSheet).Select
                 'Cells(Total + 4, 1).Value = Total
                 'Cells(Total + 4, 2).Value = x1
                 'Cells(Total + 4, 3).Value = x2
                 'Cells(Total + 4, 4).Value = x3
                 'Cells(Total + 4, 5).Value = x4
                 'Cells(Total + 4, 6).Value = x5
                 'Cells(Total + 4, 7).Value = x6
                 'Cells(Total + 4, 8).Value = x7
                 'Cells(Total + 4, 9).Value = x8
line7:            Next x8
line8:           Next x7
line9:          Next x6
line10:        Next x5
line11:      Next x4
line12:    Next x3
line13:  Next x2
      Next x1

Endtime = Timer 'calculate processing time
Totaltime = Endtime - Starttime

Sheets(OutputSheet).Select 'output results
Cells(1, 1).Value = "270 Degree Backtracking Filter Statistics " & Trial
Cells(2, 1).Value = "Total"
Cells(3, 1).Value = Total
Cells(2, 2).Value = "Processing Time "
```

```
Cells(3, 2).Value = Totaltime

Application.WindowState = xlNormal
MsgBox "The total number or routes generated is " & Total, vbOKOnly, "Route

Generator"

End Sub
```

Appendix D.

Appendix D contains the Microsoft Excel Visual Basic macro used to find the great circle distance matrices.

```
Sub GenerateDistanceMatrix()

' This macro creates the distance matrix for a list of targets.
' In order to this it calculates the great circle distance between
' pairs of targets in the list and outputs the results to a
' worksheet in a matrix arrangement.

' GenDisMatrix Macro
' Macro recorded 11/17/97 by greg ehlers
'
' Keyboard Shortcut: Ctrl+Shift+G
'
  Dim x1 As Integer 'Looping variable for the 1st target
  Dim x2 As Integer 'Looping variable for the 2d target

  Dim Lat1 As Double 'variable to hold 1st targets latitude
  Dim Lat2 As Double 'variable to hold 2d targets latitude

  Dim Long1 As Double 'variable to hold 1st target longitude
  Dim Long2 As Double 'variable to hold 2d target longitude

  Dim GCdistance As Double 'variable for great circle distance
  Dim NumberAim As Integer 'variable for the total number of targets
  Dim OutputSheet As String 'output variable worksheet for matrix
  Dim InputSheet As String 'input variable for location of target list

  Crad = 3.14159 / 180 'constant to convert radians to degrees
  Rad = 180 / 3.14159  'constant to convert degrees to radians

  InputSheet = InputBox("Select input sheet", "Route Generator", _
   "Sheet1") 'location of target list

  OutputSheet = InputBox("Select output sheet", "Route Generator", _
   "Sheet1") 'location of distance matrix

  For x1 = 1 To 10
   For x2 = 1 To 10
   Sheets(InputSheet).Select
   Lat1 = Cells(x1 + 1, 2).Value
```

```
    Long1 = Cells(x1 + 1, 3).Value
    Lat2 = Cells(x2 + 1, 2).Value
    Long2 = Cells(x2 + 1, 3).Value
    If Lat1 = Lat2 And Long1 = Long2 Then
    GCdistance = 0 'for distance between the same target
    Sheets(OutputSheet).Select
    Cells(x1 + 1, x2 + 1).Value = GCdistance
    Else
    X = Sin(Lat1 * Crad) * Sin(Lat2 * Crad) + Cos(Lat1 * Crad) * Cos(Lat2 * Crad) *
Cos((Long1 - Long2) * Crad)
    GCdistance = 60 * (Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)) * Rad
    'algebraic approximation of AMC formula for calculating
    'great circle distance.
    Sheets(OutputSheet).Select
    Cells(x1 + 1, x2 + 1).Value = GCdistance
    End If
    Next x2
Next x1

MsgBox "Distance Matrix is complete.", vbExclamation, "Route Generator"

End Sub
```

Appendix E.

Appendix E. contains the Microsoft Excel table with the location data for the eight target test problem.

Table E-1. Target Data Summary for Eight Target Test Problem

		Eight Target Problem				
		Target	Latitude	Longitude		
		1	69.19028	116.0064		
		2	59.51168	130.7303		
		3	68.4009	133.5914		
		4	69.42945	112.9425		
		5	50.88857	119.9002		
		6	60.88771	130.965		
		7	62.46756	118.8653		
		8	58.14122	127.2959		
		Minimum	50.88857	112.9425		
		Maximum	69.42945	133.5914		
		Range	18.54088	20.64892		

Appendix F.

Appendix F. contains the Microsoft Excel tables for the targets that were used for the ten trials of the ten target sample problem.

Table F-1. Target Data Summary for Ten Target Test Problems

Trial One				**Trial Six**		
Target	Latitude	Longitude		Target	Latitude	Longitude
1	58.79898	59.31209		1	53.40696	118.7436
2	64.73378	57.67332		2	48.93332	136.9159
3	69.71592	39.27617		3	69.52534	110.279
4	49.27358	152.999		4	62.42869	40.21278
5	67.52795	95.33365		5	59.60147	101.1676
6	54.94187	77.33911		6	69.33511	122.0069
7	58.21857	36.41192		7	59.79278	80.99463
8	60.52695	46.75099		8	44.01135	120.9632
9	52.72249	127.1487		9	56.48545	30.09713
10	64.10916	113.6388		10	67.19516	53.38564
Minimum	49.27358	36.41192		Maximum	44.01135	30.09713
Maximum	69.71592	152.999		Minimum	69.52534	136.9159
Range	20.44234	116.587		Range	25.51399	106.8188
Trial Two				**Trial Seven**		
Target	Latitude	Longitude		Target	Latitude	Longitude
1	58.33991	103.0467		1	49.7927	34.92567
2	42.4051	102.8544		2	54.46643	96.49274
3	44.74193	80.51146		3	66.86376	47.94511
4	40.99637	85.8443		4	68.68932	118.287
5	48.86229	95.246		5	58.54944	147.8274
6	41.25636	60.80012		6	69.76691	112.5691
7	68.64991	57.48973		7	42.93073	46.05491
8	48.58059	61.42549		8	54.40865	113.3169
9	66.80184	83.43332		9	66.46399	34.97273
10	41.98615	106.865		10	54.58251	56.22602
Minimum	40.99637	57.48973		Minimum	42.93073	34.92567
Maximum	68.64991	106.865		Maximum	69.76691	147.8274
Range	27.65354	49.37526		Range	26.83618	112.9017
Trial Three				**Trial Eight**		
Target	Latitude	Longitude		Target	Latitude	Longitude
1	56.65338	35.74938		1	40.19409	82.46852
2	61.71215	89.66568		2	57.84602	91.34506

3	44.06042	124.7755		3	65.32408	151.2455
4	68.50784	39.42125		4	57.04731	68.00798
5	68.49973	129.4192		5	53.01318	51.53945
6	66.8401	143.5952		6	51.34494	72.47955
7	54.42316	148.2889		7	40.74264	75.25463
8	46.50555	131.3553		8	45.731	46.12233
9	50.60758	44.78755		9	66.31579	44.265
10	55.2731	130.4459		10	56.53467	32.06946
Minimum	44.06042	35.74938		Minimum	40.19409	32.06946
Maximum	68.50784	148.2889		Maximum	66.31579	151.2455
Range	24.44742	112.5395		Range	26.1217	119.1761
Trial Four				**Trial Nine**		
Target	Latitude	Longitude		Target	Latitude	Longitude
1	67.88061	34.00706		1	48.82704	68.82505
2	40.11858	104.0315		2	45.57811	57.46632
3	57.41276	104.3303		3	53.28174	51.31454
4	48.52976	102.9233		4	67.66608	41.24757
5	43.35205	46.43896		5	63.96357	44.64783
6	48.57117	132.1637		6	40.4243	32.02264
7	52.68478	98.80842		7	68.71644	146.2302
8	64.68278	150.0936		8	42.78184	121.0951
9	62.97642	44.87243		9	49.04029	68.34861
10	59.65045	89.61182		10	40.10411	56.81529
Minimum	40.11858	34.00706		Minimum	40.10411	32.02264
Maximum	67.88061	150.0936		Maximum	68.71644	146.2302
Range	27.76203	116.0866		Range	28.61232	114.2076
Trial Five				**Trial Ten**		
Target	Latitude	Longitude		Target	Latitude	Longitude
1	53.89258	158.2693		1	64.76038	88.0124
2	51.26497	70.77195		2	47.47552	109.5942
3	55.78689	115.859		3	48.31301	143.3091
4	44.34513	127.2774		4	60.65775	92.33907
5	52.37309	73.76399		5	51.97516	154.9195
6	44.28915	32.99064		6	65.01094	102.8247
7	57.26752	95.87184		7	62.64234	78.05729
8	44.19803	148.4962		8	66.86944	126.9432
9	61.47959	105.0337		9	43.14184	109.0364
10	57.80684	81.13936		10	40.59687	145.7148
Minimum	44.19803	32.99064		Minimum	40.59687	78.05729
Maximum	61.47959	158.2693		Maximum	66.86944	154.9195
Range	17.28155	125.2786		Range	26.27256	76.86221

Bibliography

1. Brandao, Jose and Mercer, Alan, "ATabu Search Algorithm for the Multi-trip Vehicle Routing and Scheduling Problem," *European Journal of Operational Research*, 100: 180-191 (1997).

2. Gallagher, Mark A. *Weapons Allocation Model*, WAM overview used at the 64th MORS Symposium, 19 June, 1996.

3. Glover, Fred, Taillard, Eric and De Werra, Dominique, "A User's Guide to Tabu Search," *Annals of Operations Research*, 41: 3-28 (1993).

4. Fair, Elliot T., *Modeling MIRV Footprint Constraints in the Weapon Assignment Model*. MS thesis, AFIT/GOA/ENS/97M-04. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1997.

5. Fisher, Marshall L. and Jaikumar, Ramchandran, "A Generalized Assignment Heuristic for Vehicle Routing," *Networks*, 11: 109-124 (1981).

6. Laporte, Gilbert and Osman, Ibrahim H., "Routing Problems: A Bibliography," *Annals of Operations Research*, 61: 227-262 (1995).

7. Microsoft Corporation. *Visual Basic User's Guide: Automating, Customizing, and Programming in Microsoft Excel with Microsoft Visual Basic Programming System, Application Edition.* 1993-1994.

8. Mitchell, Frank E., "The Use of Preprocessed Cruise Missile Data for Strategic Planning," Report to USSTRATCOM, GDE Systems, Bellevue, NE, 1996.

9. Nemhauser, G., and Wolsey, L., Integer and Combinatorial Optimization. New York: Wiley, 1988.

10. Reeves, Colin R., *Modern Heuristic Techniques for Combinatorial Problems*. New York: Halsted Press, 1993.

11. Reid, Thomas F., Deputy Department Head, Department of Mathematics and Statistics, Air Force Institute of Technology, Personal Interview, 8 January 1998.

12. Renaud, Jacques, Laporte, Gilbert, and Boctor, Fayes F., "A Tabu Search Heuristic for the Multi-depot Vehicle Routing Problem," *Computers and Operations Research*, 23: 229-235, (1996)

13. Rendall, David, *Jane's Aircraft Recognition Guide*. Glascow: Harper Collins Publishers, 1996.

14. Taillard, E., " Parallel Iterative Search Methods for Vehicle Routing Problems," *Networks*, 23: 661-673, (1993).

15. Volgenant, Ton and Jonker, Roy, "The Symmetric Traveling Salesman Problem and Edge Exchanges in Minimal 1-trees," *European Journal of Operations Research*, 12: 394-403, (1983)

16. Wackerly, Dennis D. and others, *Mathematical Statistics with Applications*. New York: Duxbury Press, 1996.

Vita

Captain Gregory J. Ehlers was born 29 December 1963 in Omaha NE. He graduated from Millard South High School in 1982 and attended Creighton University in Omaha. In 1984, he enlisted in the United States Air Force and served as an Airborne Cryptologic Russian Linguist from 1984 until 1991. After completing his Bachelor's of Science in Mathematics at Creighton, he was commissioned at the Officer Training School on Lackland AFB, TX on 23 September 1992. He served as a Fleet Satellite (FLTSAT) Communications Operator, Satellite Engineer Officer, and Chief FLTSAT Procedures Section at 3d Space Operations Squadron, Falcon AFB, CO. He entered the Graduate School of Engineering, Air Force Institute of Technology, in August 1996.

Permanent Address: 1012 Eldorado Dr.

Omaha, NE 68154

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1997	Master's Thesis

4. TITLE AND SUBTITLE
Generating Bomber Routes for the Delivery of Gravity Weapons

5. FUNDING NUMBERS

6. AUTHOR(S)
Gregory J. Ehlers, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology/ENS
2750 P Street
Wright-Patterson AFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GOR/ENS/98M-11

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Major Mark Gallagher
USSTRATCOM/J533
901 SAC BLVD
Offutt AFB, NE 68133-6500

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
James T. Moore, LtCol, USAF
937-255-6565 Ext. 4337
jtmoore@afit.af.mil

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

United States Strategic Command (USSTRATCOM) is updating the Arsenal Exchange Model (AEM) that it uses for allocating strategic weapons to targets. AEM does not model the range constraints on bomber routes. These routes are used to deliver gravity munitions. In order to include these constraints in the new method, the Weapons Allocation Model (WAM), a pool of routes that satisfy the range constraints needs to be created. The number of possible routes to consider is extremely large and it is not possible to enumerate them all. Therefore, the method to form the pool of routes needs to reduce the number of routes considered and selected to a manageable number. This research examines several methods for generating routes in an attempt to find the best method. All of these methods use implicit enumeration to create a pool of routes. The best method uses a combination of distance and flight direction to restrict the number of targets available to generate a route. However, the reduction provided by this method is not large enough and preprocessing the target database is also used. The method is tested against sample problems of several sizes to find the best way to use the method to generate routes in a problem of comparable size to the USSTRATCOM problem.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Vehicle Routing Problem, Bomber Routes, Implicit Enumeration, Arsenal Exchange Model, Weapon Assignment Model;	95
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL